



Course on Computer Communication and Networks

EDA344/ DIT 423 / LEU062

Lecture 1

Chapter 1: Introduction

Part A: Internet, Protocol Layering

Lecturer: Marina Papatriantafilou

Based on the book Computer Networking: A Top Down Approach, Jim Kurose, Keith Ross, Addison-Wesley.

Leonard Kleinrock about the Internet



AN INTERVIEW WITH...

Leonard Kleinrock

Leonard Kleinrock is a professor of computer science at the University of California, Los Angeles. In 1969, his computer at UCLA became the first node of the Internet. His creation of packet-switching principles in 1961 became the technology behind the Internet. He received his B.E.E. from the City College of New York (CCNY) and his masters and PhD in electrical engineering from MIT.

What was going through your mind when you sent the first host-to-host message [from UCLA to the Stanford Research Institute]?

Frankly, we had no idea of the importance of that event. We had not prepared a special message of historic significance, as did so many inventors of the past (Samuel Morse with "What hath God wrought," or Alexander Graham Bell with "Watson, come here! I want you," or Neal Armstrong with "That's one small step for a man, one giant leap for mankind.") Those guys were *smart!* They understood media and public relations. All we wanted to do was to log in to the SRI computer. So we typed the "L", which was correctly received, we typed the "o" which was received, and then we typed the "g" which caused the SRI host computer to crash! So, it turned out that our message was the shortest and perhaps the most prophetic message ever, namely "Lo!" as in "Lo and behold!"

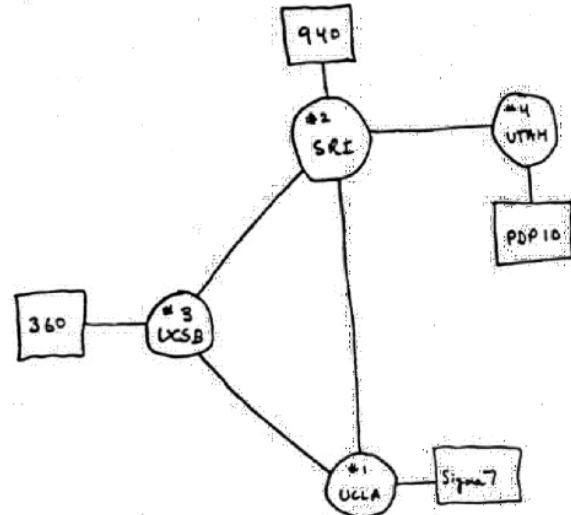
Earlier that year, I was quoted in a UCLA press release saying that once the network was up and running, it would be possible to gain access to computer utilities from our homes and offices as easily as we gain access to electricity and telephone connectivity. So my vision at that time was that the Internet would be ubiquitous, always on, always available, anyone with any device could connect from any location, and it would be invisible. However, I never anticipated that my 99-year-old mother would use the Internet—and indeed she did!

And a little more

<https://www.youtube.com/watch?v=Zc1tZ8JsZvg>

Stories...

1969

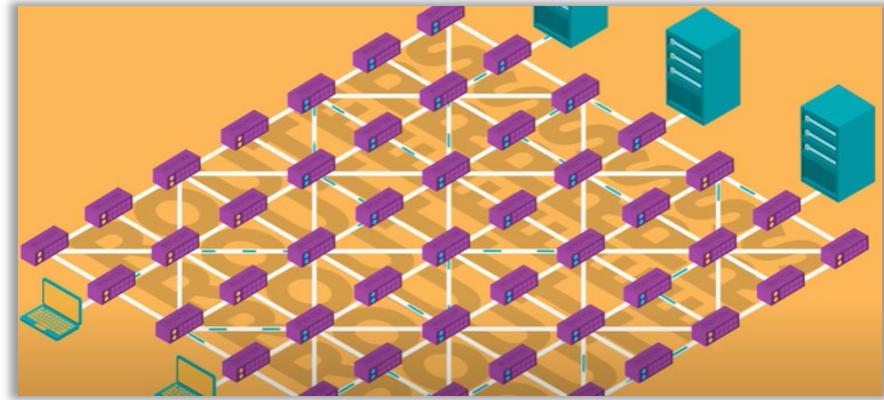
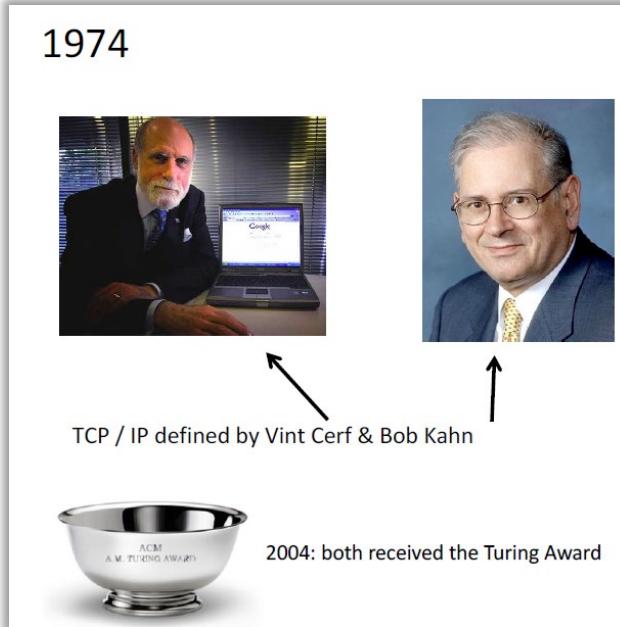
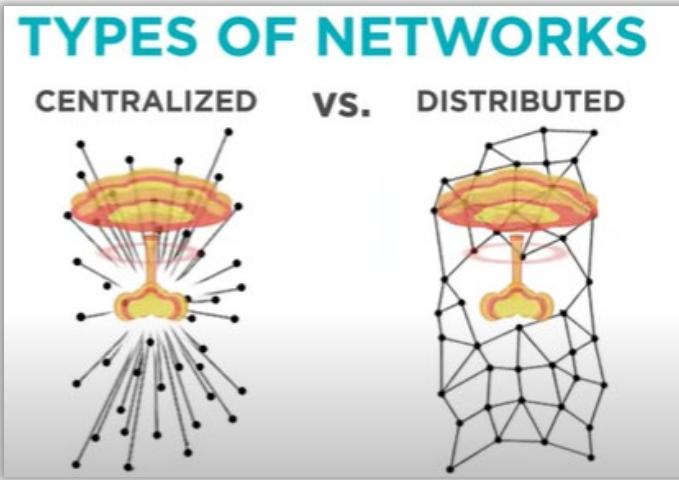


ARPANET begins...with a deployment at UCLA, Stanford, UCSB, and Utah (one computer per site)

60s

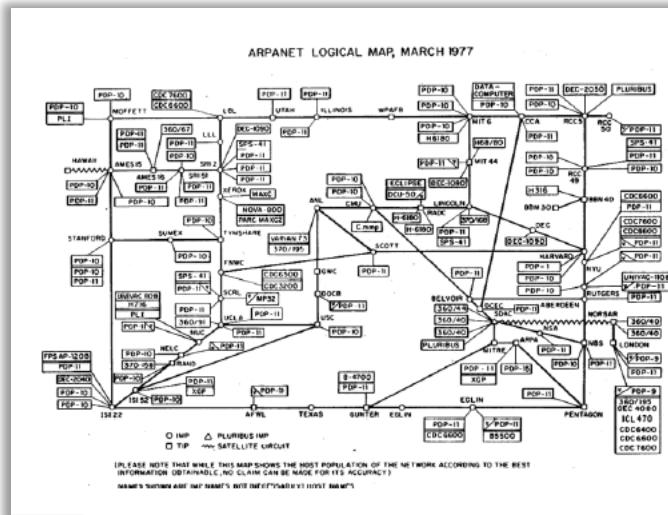
- Data-packet networking

Stories (cont)



- Data-packet networking
- University contributions form the de-facto standards!

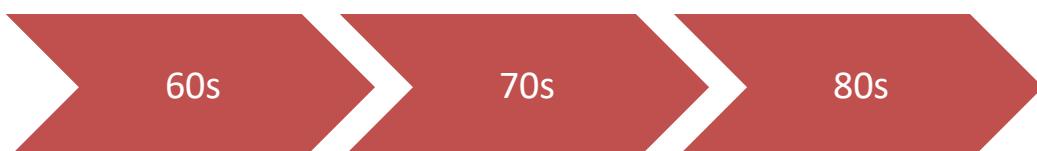
Stories (cont)



1984



Paul Mockapetris introduces DNS



- Data-packet networking
- University contributions form the de-facto standards!
- Global networking
- WWW emerges

1989 – The Web Emerges

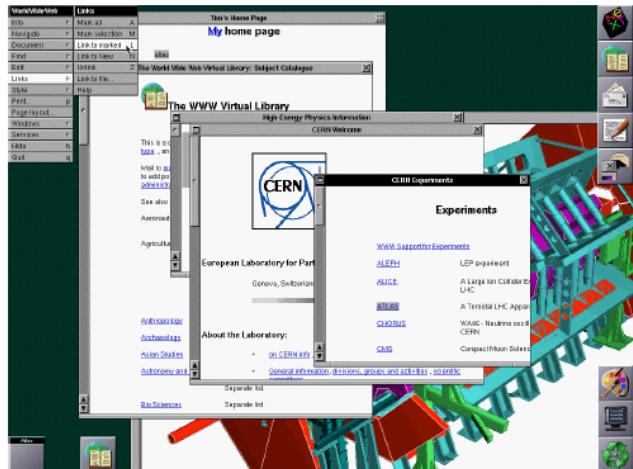
A screenshot of a computer screen displaying the original proposal for the World Wide Web. The title "Information Management: A Proposal" is at the top. Below it is a photograph of Tim Berners-Lee sitting in a chair. To his right is a flowchart titled "Overview" that illustrates the proposed system for managing information. The flowchart shows various components like "Information Sources", "User Requests", and "Information Servers", connected by arrows indicating the flow of data and requests. The entire document is presented in a classic Microsoft Word-like interface.

Tim Berners-Lee writes “Information Management: A proposal” at CERN

2016 Turing award

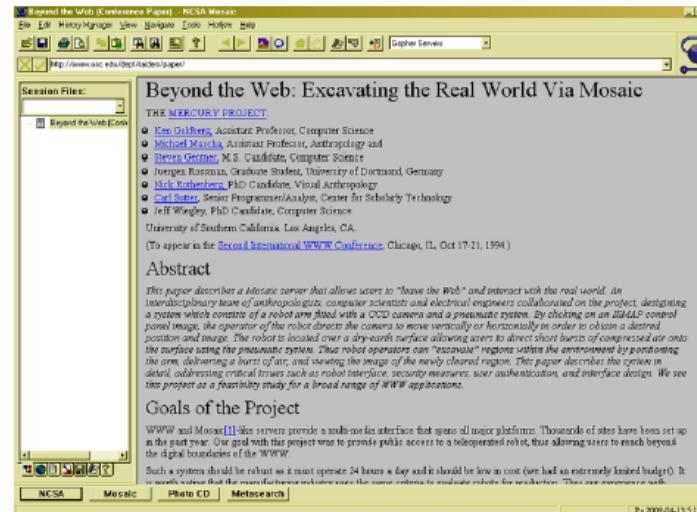
Stories (cont)

1990



First browser developed at CERN

1993



Mosaic became the first graphical browser

CERN agrees to allow public use of web protocol royalty-free!

60s

70s

80s

90s

- Data-packet networking

- University contributions form the de-facto standards!

- Global networking
- WWW emerges

- WWW open for people and business

1995+

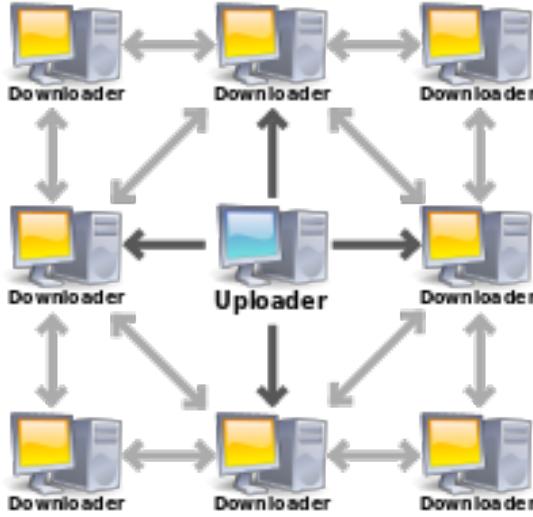
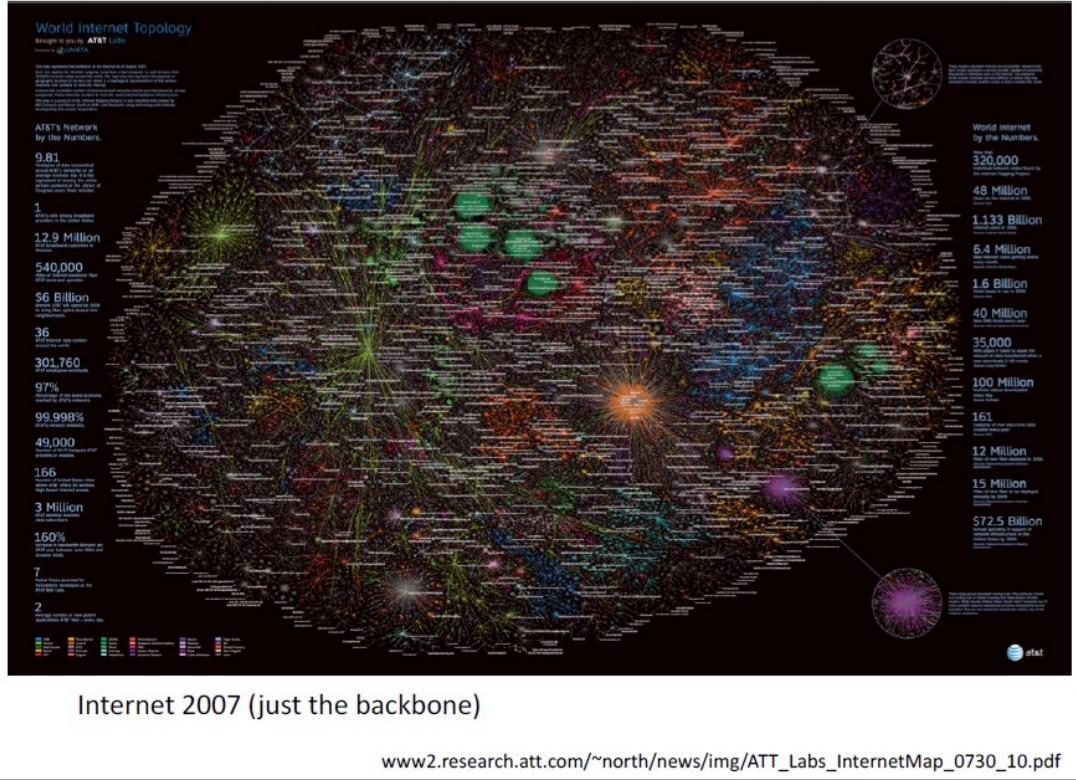
Amazon arrives and the commercialization of the web begins



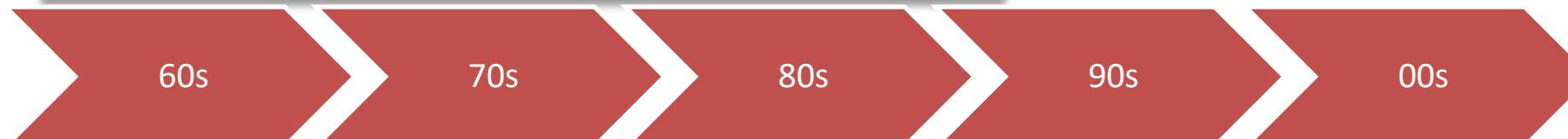
Amazon
circa
1999



Stories (cont)

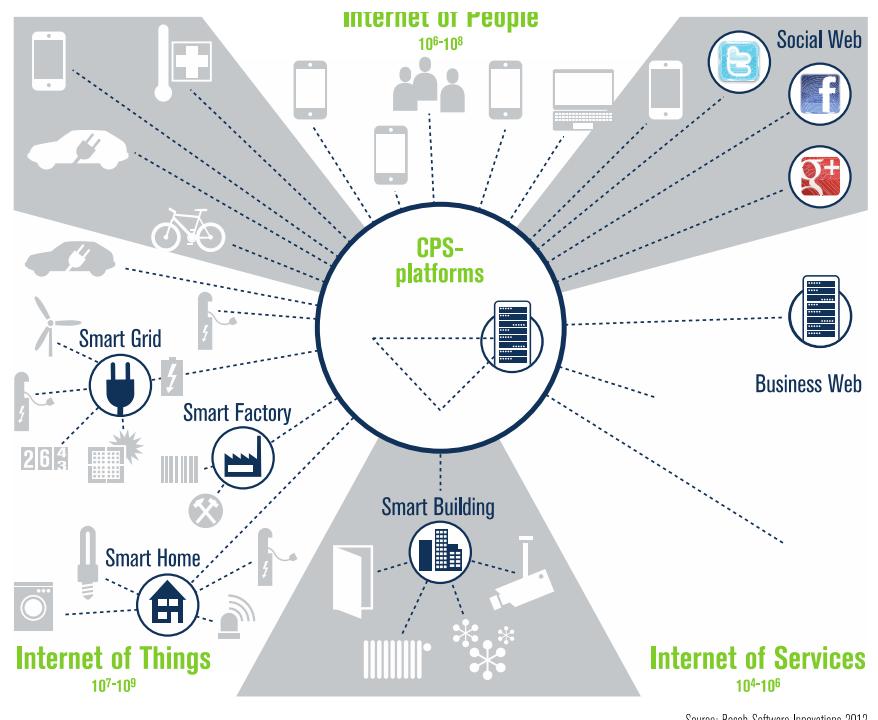


Img: wikipedia

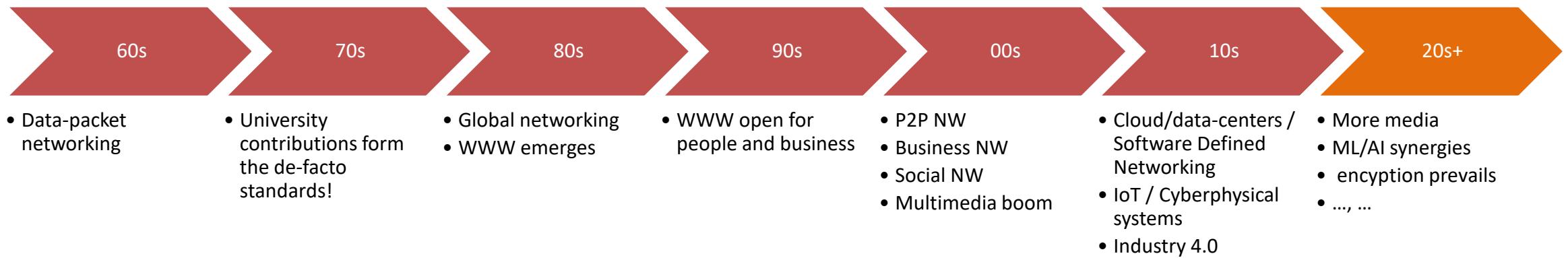


- Data-packet networking
- University contributions form the de-facto standards!
- Global networking
- WWW emerges
- WWW open for people and business
- P2P NW
- Business NW
- Social NW

Stories (cont)



Stories TBC...



Roadmap

- What's the Internet
 - Nuts&bolts view
 - Service view
 - Distinction between network edge and network core
- Layers of abstraction, protocols
 - ISO/OSI & Internet layer structure
 - Data communication through layers: physical and logical view



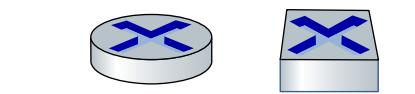


the Internet: “nuts and bolts” view (1- HW)



Billions of connected
computing devices:

- *hosts = end systems*
- running *network apps* at Internet’s “edge”



Packet switching devices:
forward packets (chunks of data)

- *routers, switches*



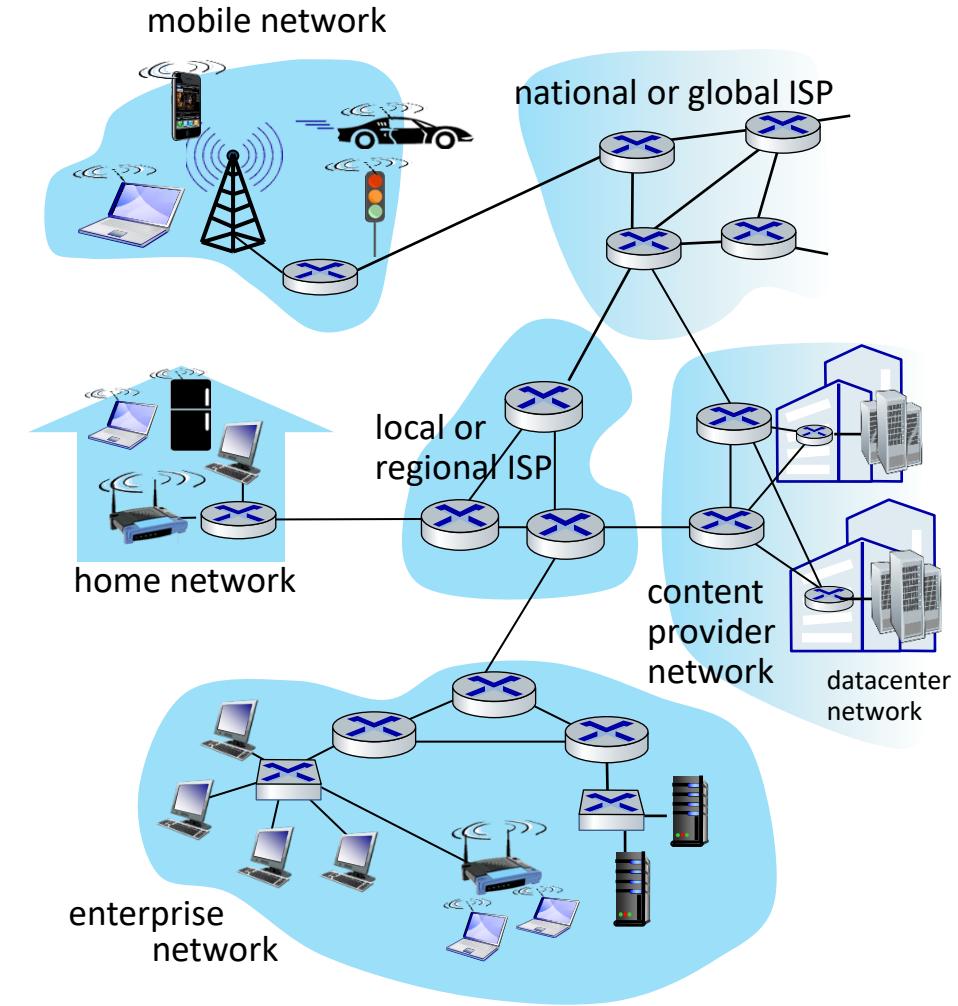
Communication links

- fiber, copper, radio, satellite
- transmission rate: *bandwidth*



Networks

- collection of devices, routers, links: managed by an organization



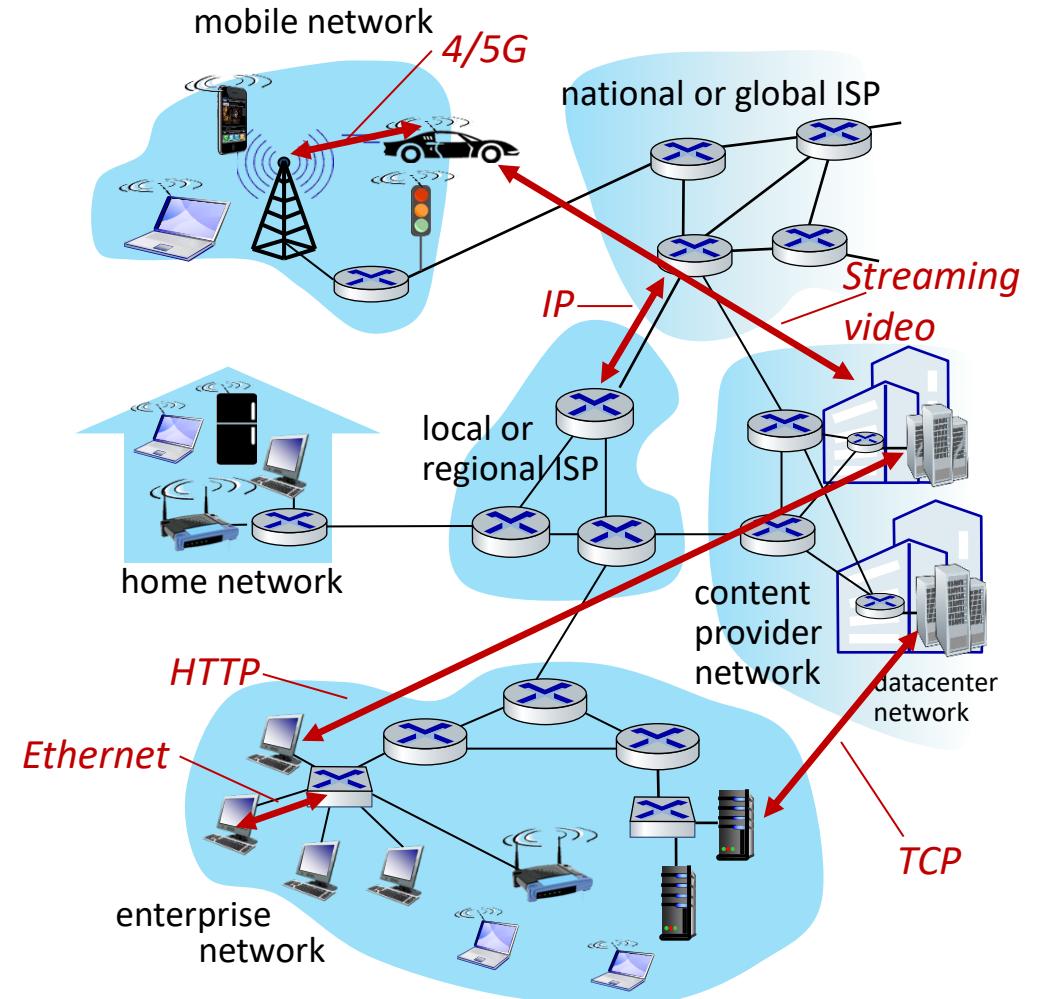
“Fun” Internet appliances in “Internet of things”





the Internet: “nuts and bolts” view (2-“SW”)

- *Internet: “network of networks”*
 - Interconnected ISPs
- *protocols are everywhere*
 - control sending, receiving of messages
 - e.g., HTTP (Web), TCP, IP, WiFi, 4G/5G, Ethernet
- *Internet standards*





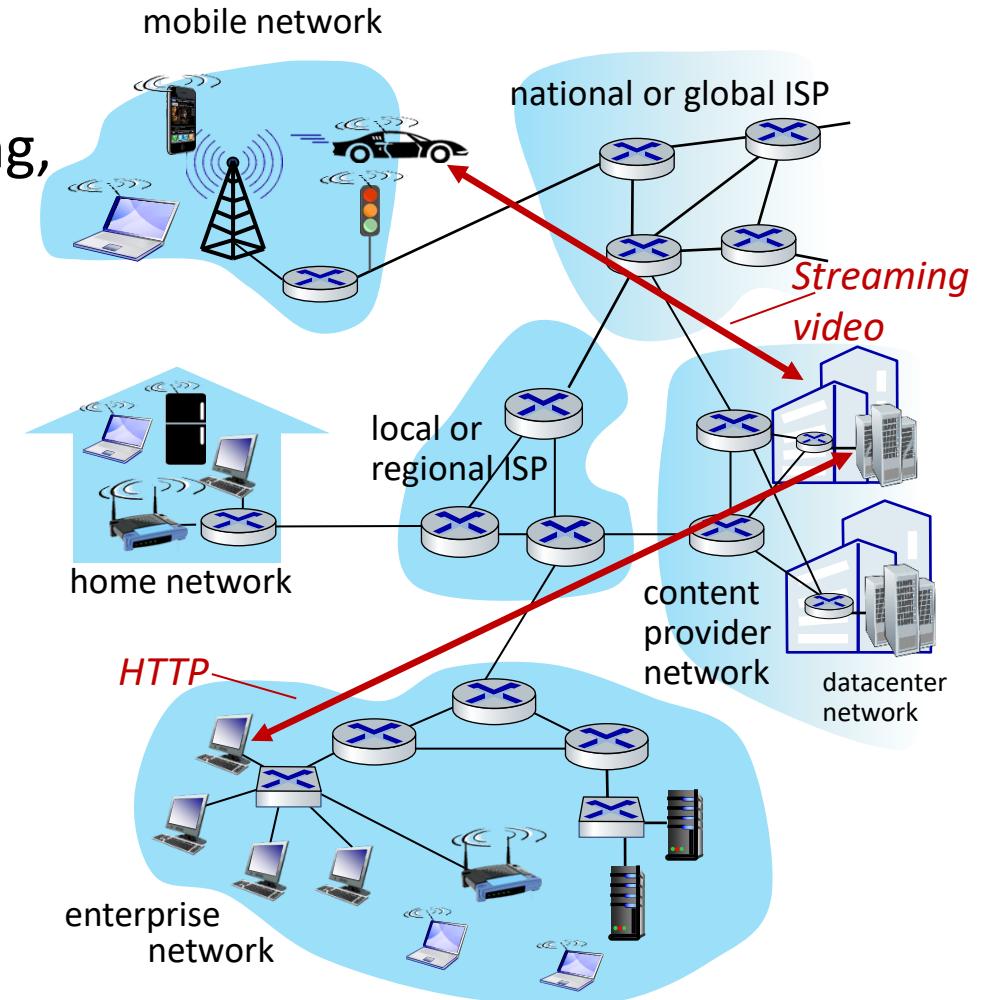
the Internet: *service view*

Communication *infrastructure* enables distributed applications:

- Web, VoIP, email, games, e-commerce, file sharing, social++ apps

Communication *services & programming interfaces* provided to apps:

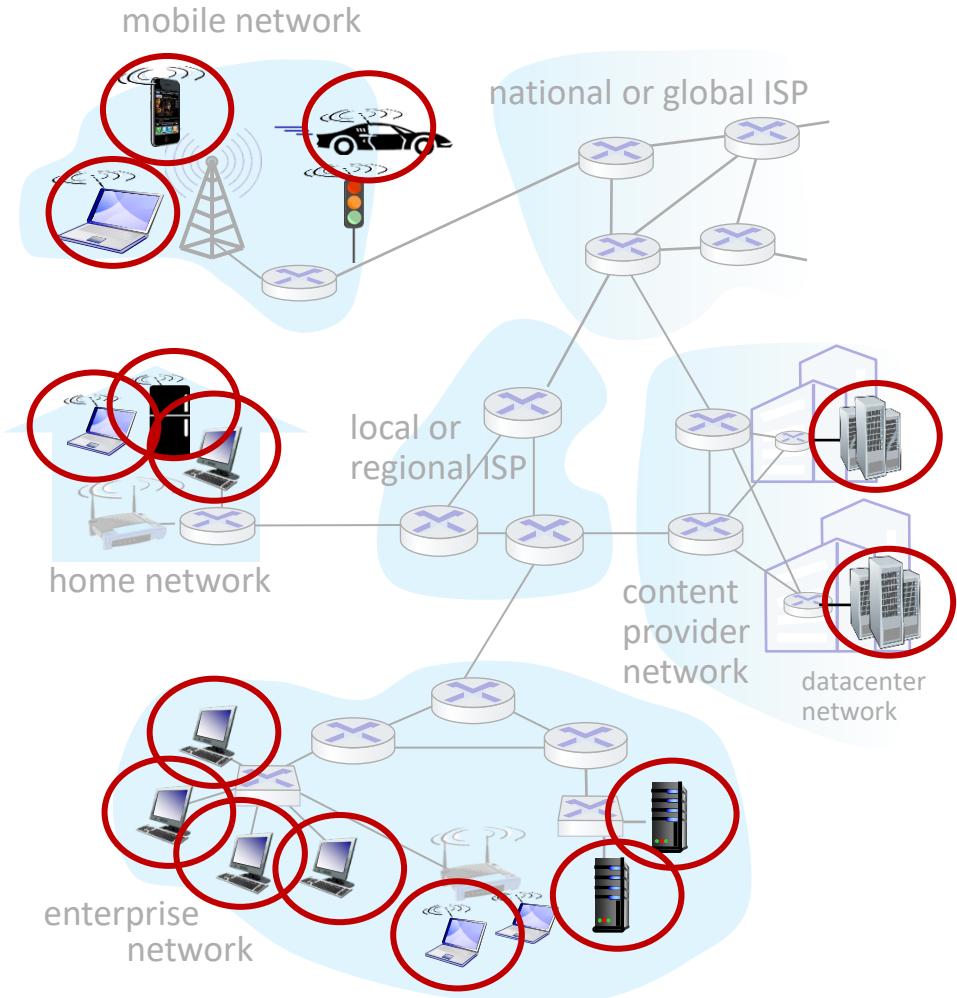
- Reliable, in-order data delivery from source to destination (aka *connection-oriented*)
- “best effort” (unreliable) data delivery (aka *connectionless*)



A closer look at Internet structure

Network edge:

- hosts: clients and servers
- servers often in data centers



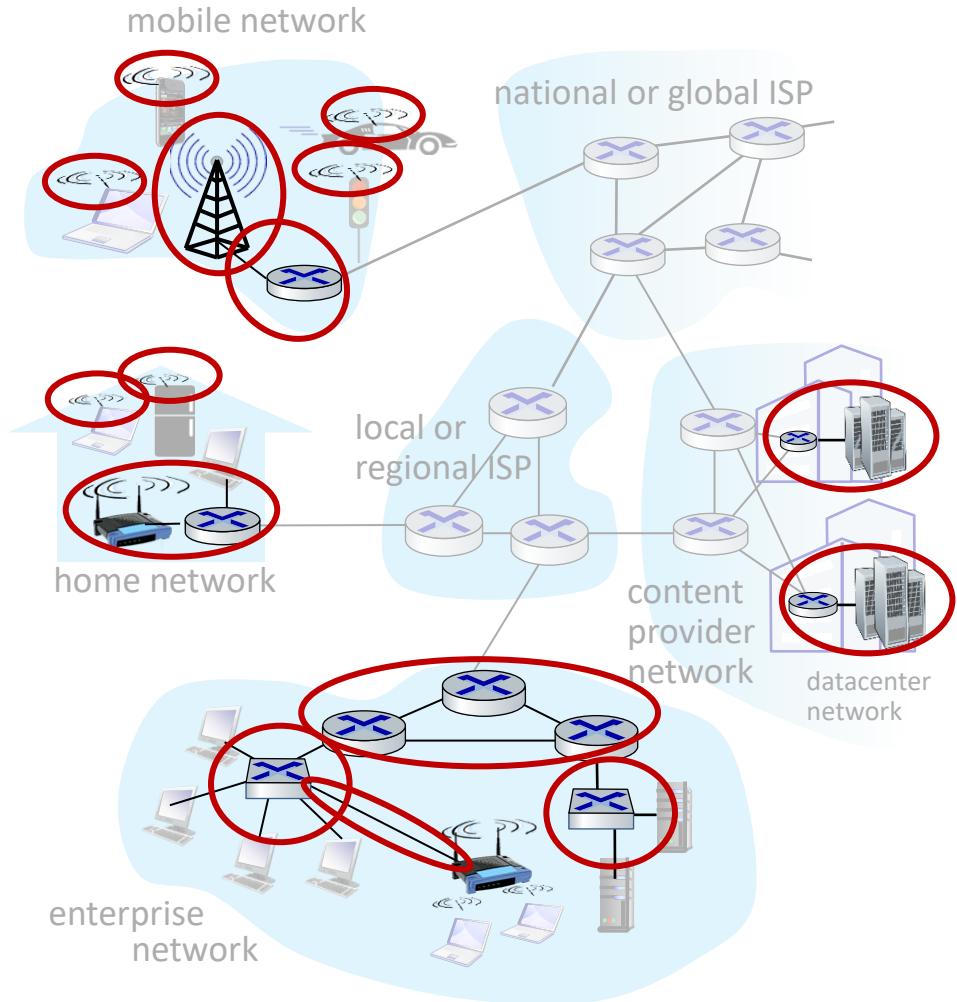
A closer look at Internet structure

Network edge:

- hosts: clients and servers
- servers often in data centers

Access networks, physical media:

- wired, wireless communication links



A closer look at Internet structure

Network edge:

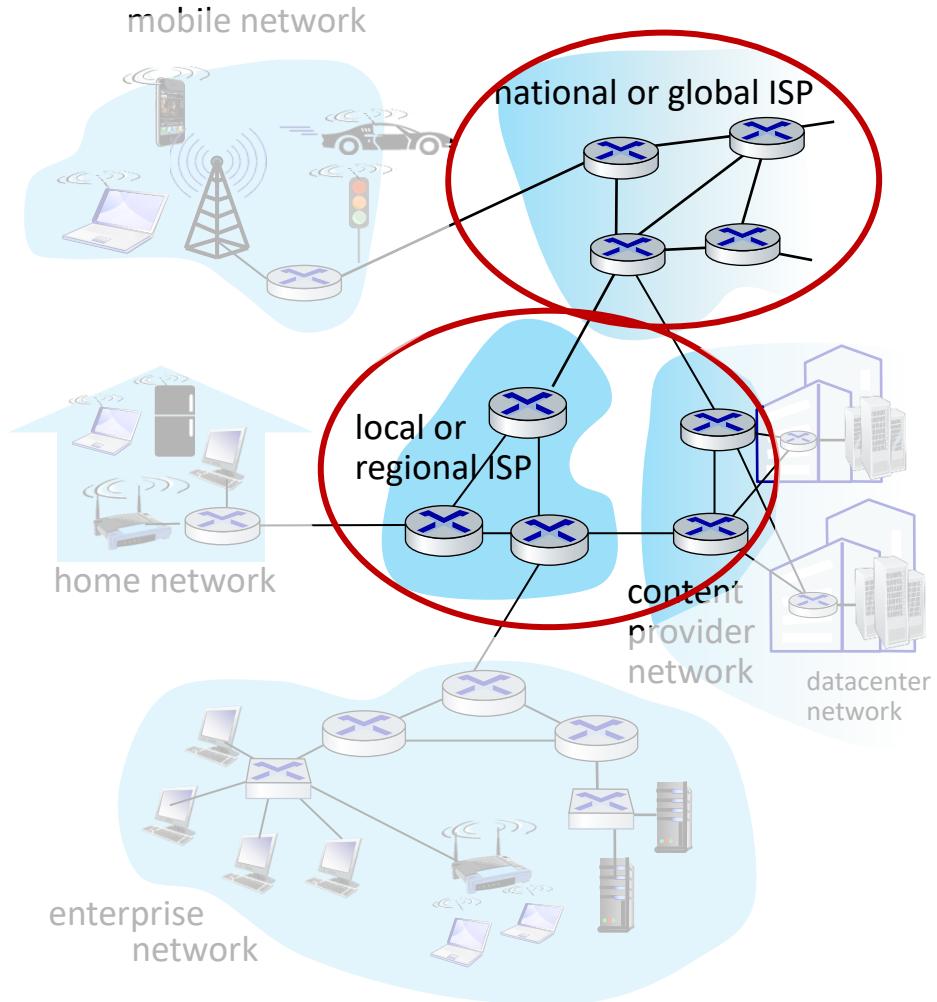
- hosts: clients and servers
- servers often in data centers

Access networks, physical media:

- wired, wireless communication links

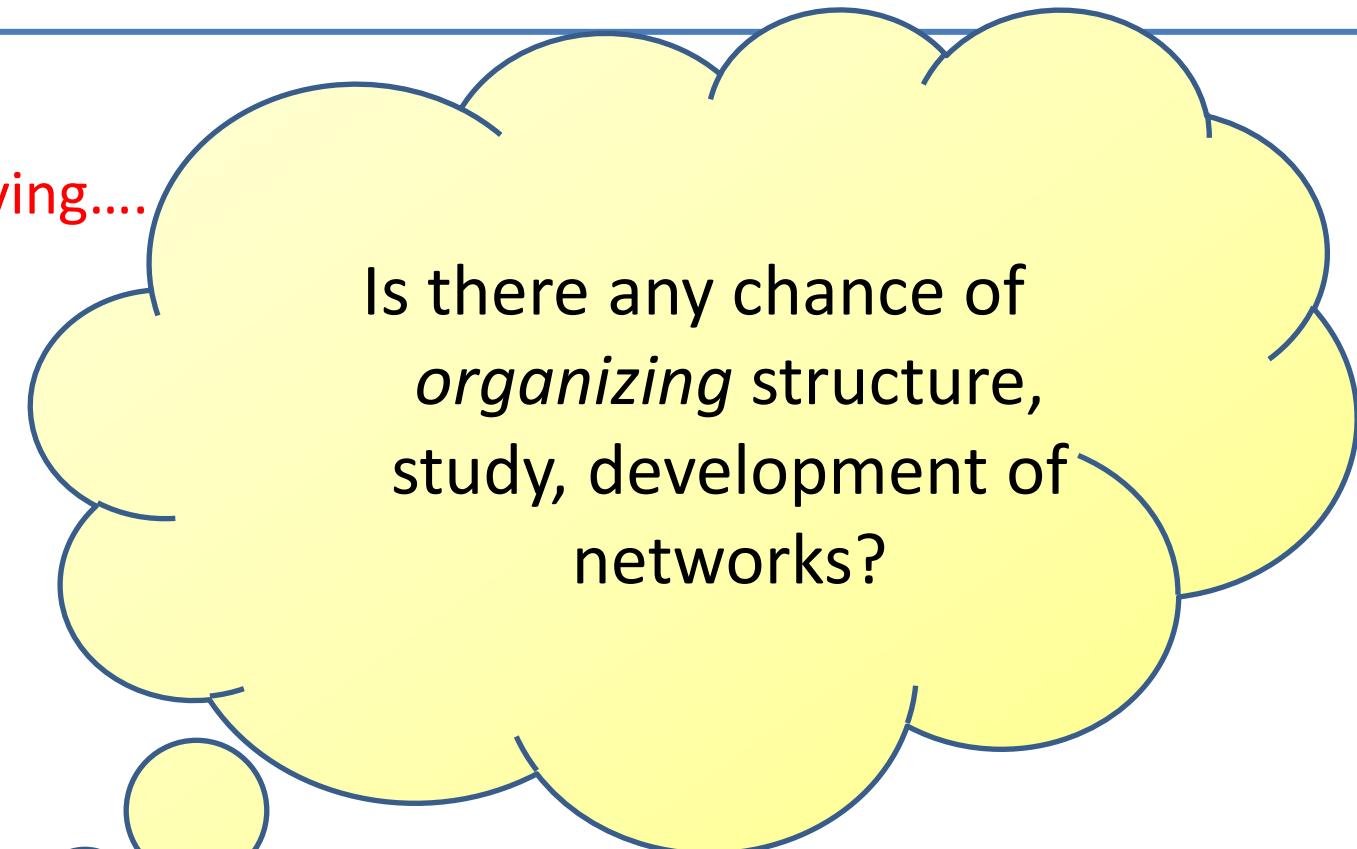
Network core:

- interconnected routers
- network of networks



Networks are complex and evolving....

- Hosts, routers, links
- Services, applications
- Hardware, software
- Networks of Networks
-



Is there any chance of
organizing structure,
study, development of
networks?



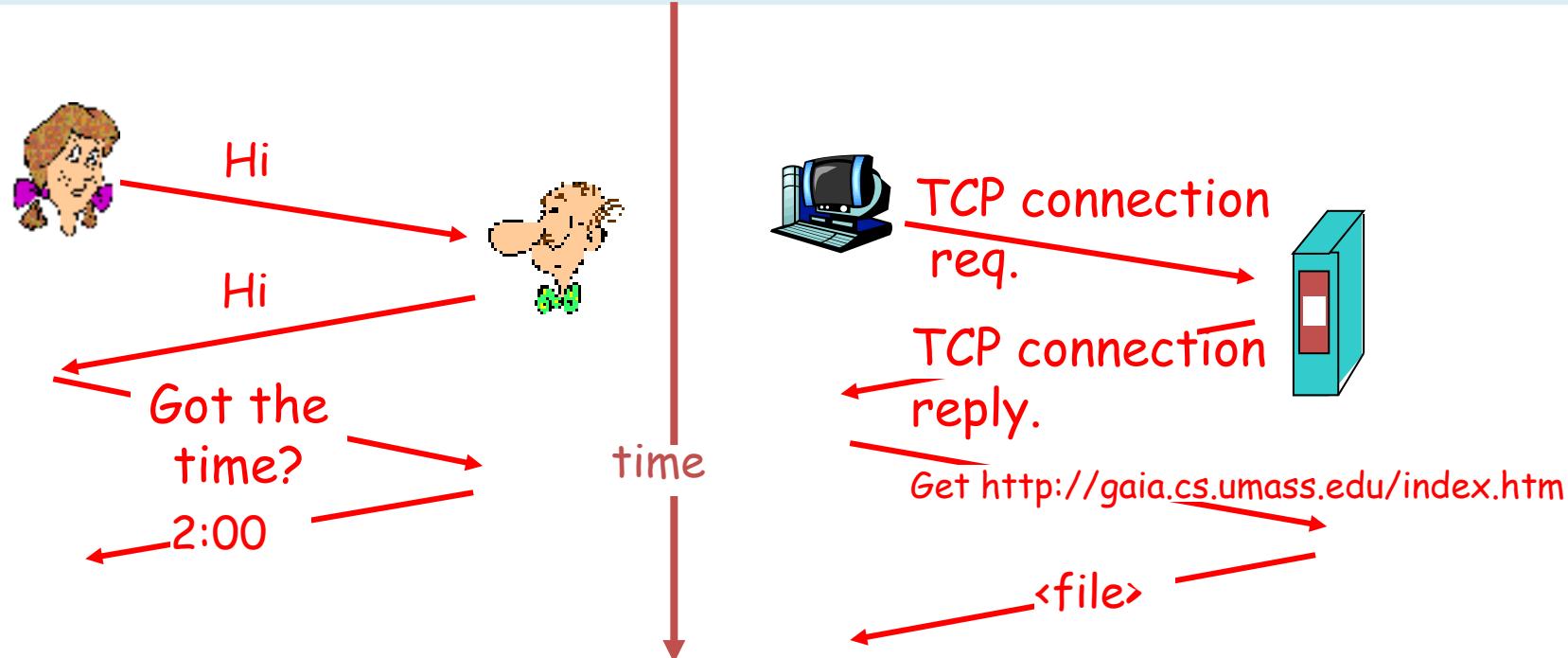
Roadmap



- What's the Internet
 - Nuts&bolts view
 - Service view
 - Distinction between network edge and network core
- Layers of abstraction, protocols
 - ISO/OSI & Internet layer structure
 - Data communication through layers: physical and logical view

What's a protocol?

Examples: a human protocol and a computer network protocol:



2-(or more)-participant interface: defines

- messages exchanged with peer entity: *format, order of msgs*
- *actions todo on msg transmission, receipt*

Terminology:

Layers, Protocols, Interfaces

Each **layer implements services**

- via its own internal actions
- relying on services by layer below

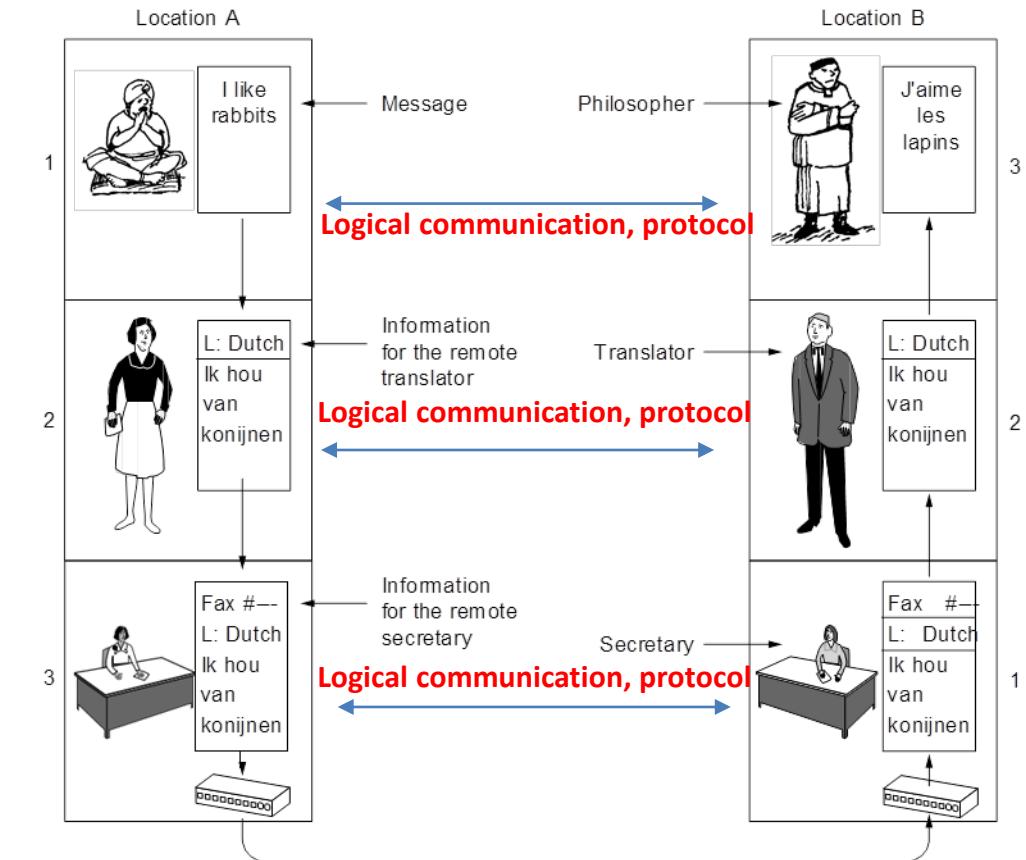
It **provides** services to the upper layer and **uses** services of the layer below

- **abstracting/shielding** from implementation details

Layer n on a host carries a conversation with layer n on another host

host-to-host interface: defines actions and message exchanges (incl. format, order) with peer entity = **logical communication, following a protocol**

System architecture: set of layers, service definitions, implemented through a **protocol stack**



The philosopher-translator-secretary architecture.

Fig. A. Tanenbaum Computer Networks

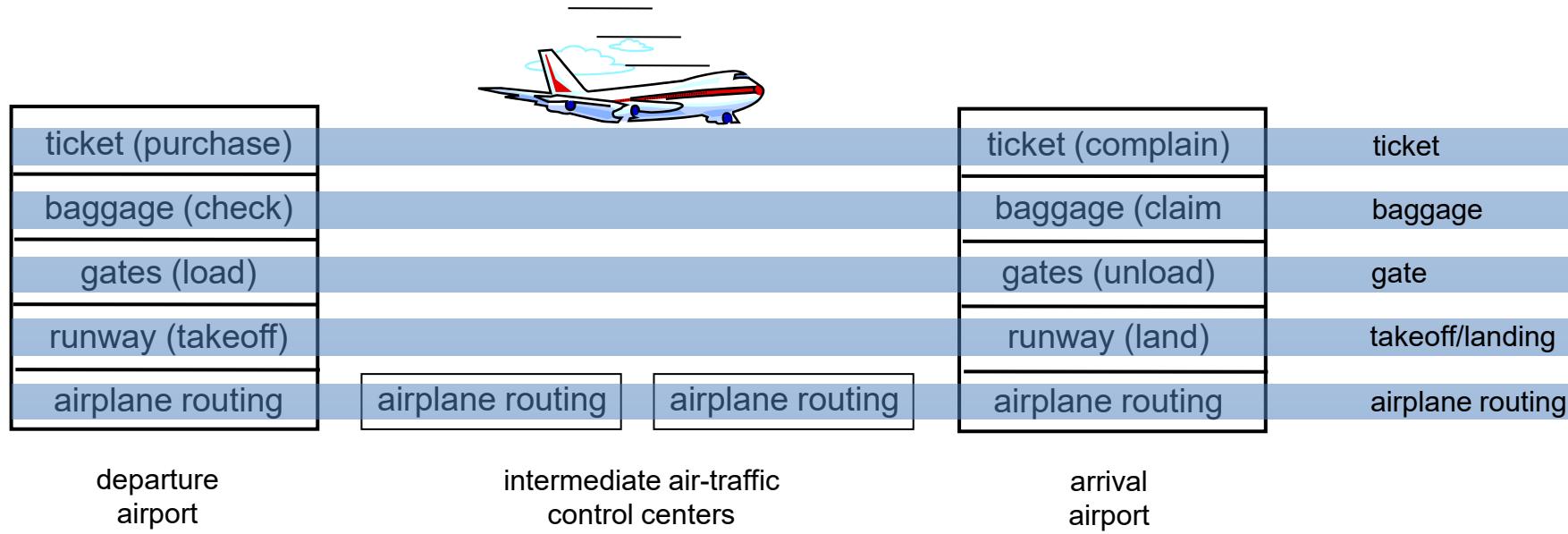
Why layering and protocols/standards?

Dealing with complex systems:

- structure allows to identify & relate complex system's pieces
 - layered **reference model** for discussion
- **modularization eases maintenance:**
 - change of implementation of layer's service transparent to rest of system
 - e.g., change of a secretary shouldn't affect rest of system

Another example:

Layering of airline functionality



Roadmap

- What's the Internet
 - Nuts&bolts view
 - Service view
 - Distinction between network edge and network core
- Layers of abstraction, protocols
 - ISO/OSI & Internet layer structure
 - Data communication through layers: physical and logical view



Layering – Some history: The OSI Reference Model

ISO (International Standards Organization) defined the OSI (Open Systems Interconnect) model to help vendors create interoperable network implementation

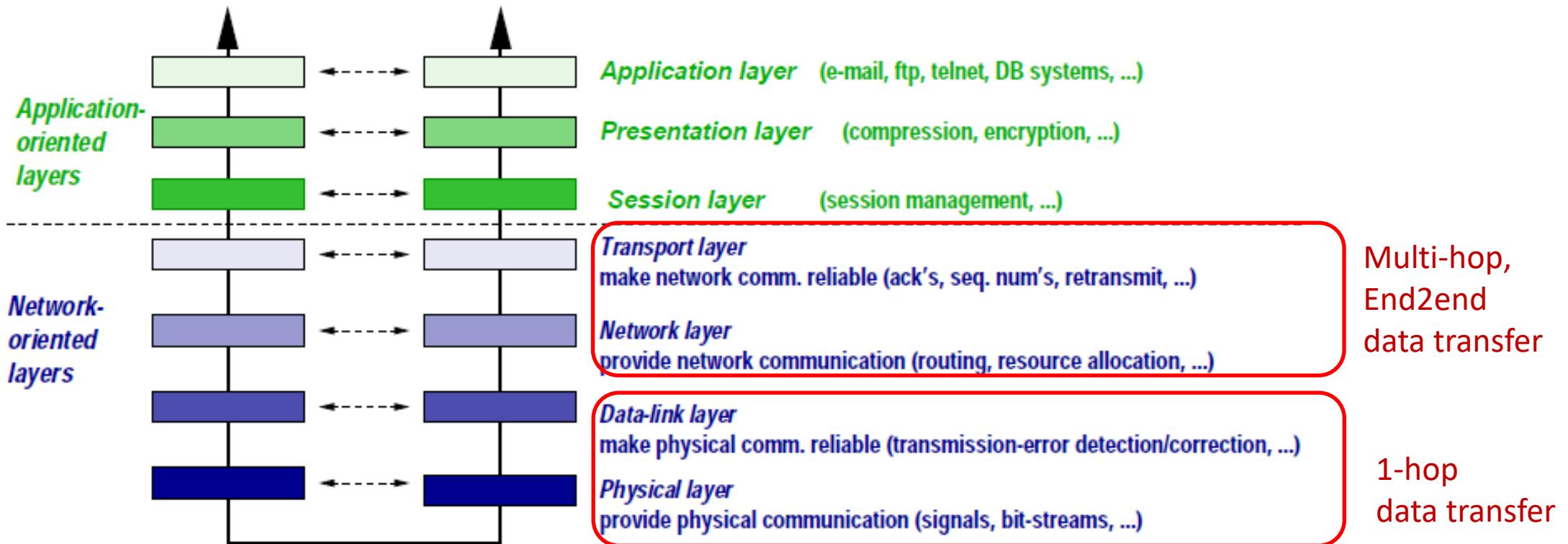


Fig. Steen, Sips : Computer and Network organization

``X dot'' series (X.25, X. 400, X.500) OSI model implementation (protocol stack)

Internet protocol stack layers&protocols

Application: protocols supporting *network applications*

http (*web*), smtp (*email*), p2p, streaming, CDN, ...

Transport: process2process (end2end) data transfer

UDP (user Datagram Prot.), TCP (Transmission Control Prot.)

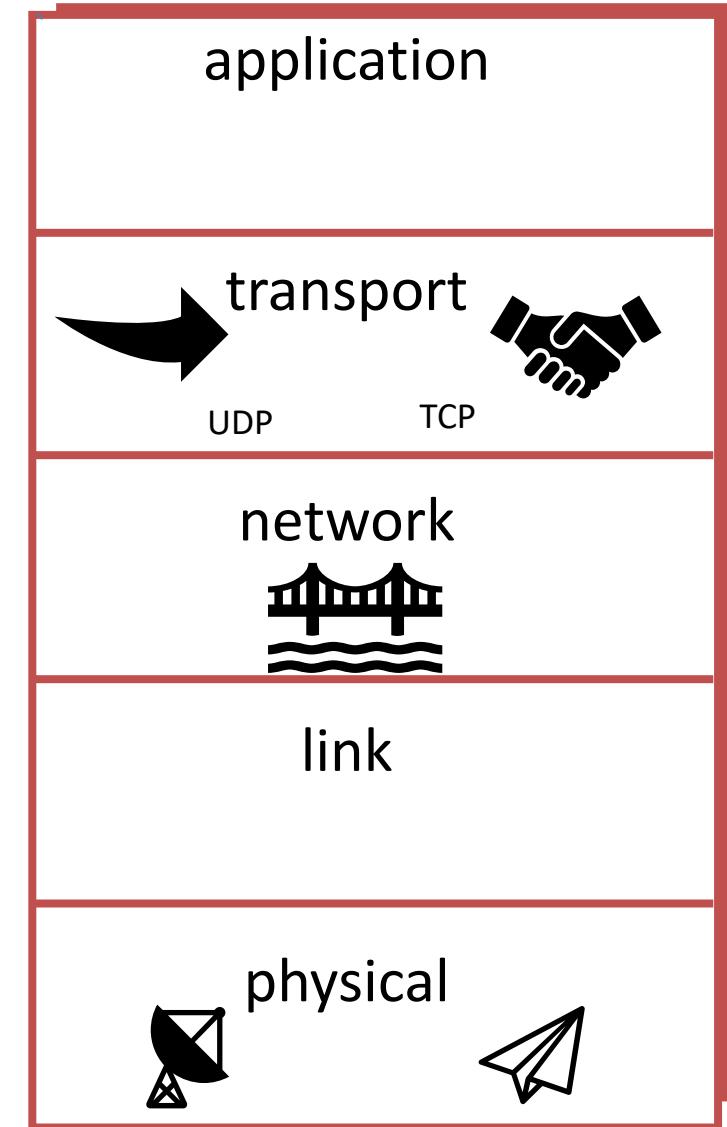
Network: routing of datagrams (independent data-packets), connecting nodes at different physical networks

**Universal addressing to bridge NWs with other addressing formats,
routing protocols**

Link: data transfer between directly connected nodes

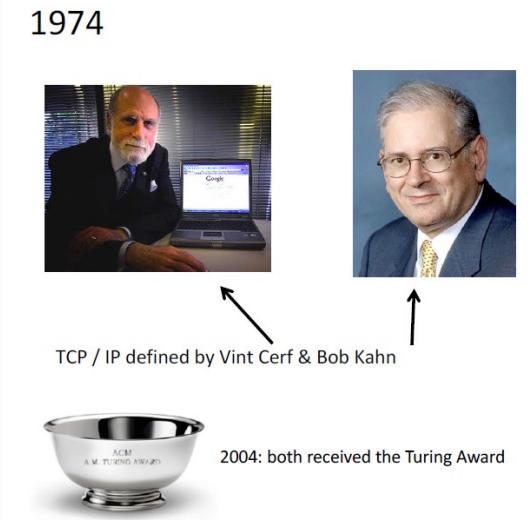
Ethernet, WiFi, ...

Physical: bit-transmission on the physical medium between directly connected nodes



Internet protocol stack

- Architecture simple, not as thoroughly thought as OSI's; Internet stack “missing”:
 - *Presentation layer*: interpret meaning of data, e.g., encryption, compression,
 - *Session layer*: synchronization, checkpointing, ...
- BUT: successful protocol suite (**de-facto standard**)
 - was there when needed (OSI implementations were too complicated)
 - did not want to change the existing NWs, rather accept them and help them connect, i.e. “speak” with each other
 - freely distributed with UNIX



Internet standards maintained through

- RFC: Request for comments
- IETF: Internet Engineering Task Force

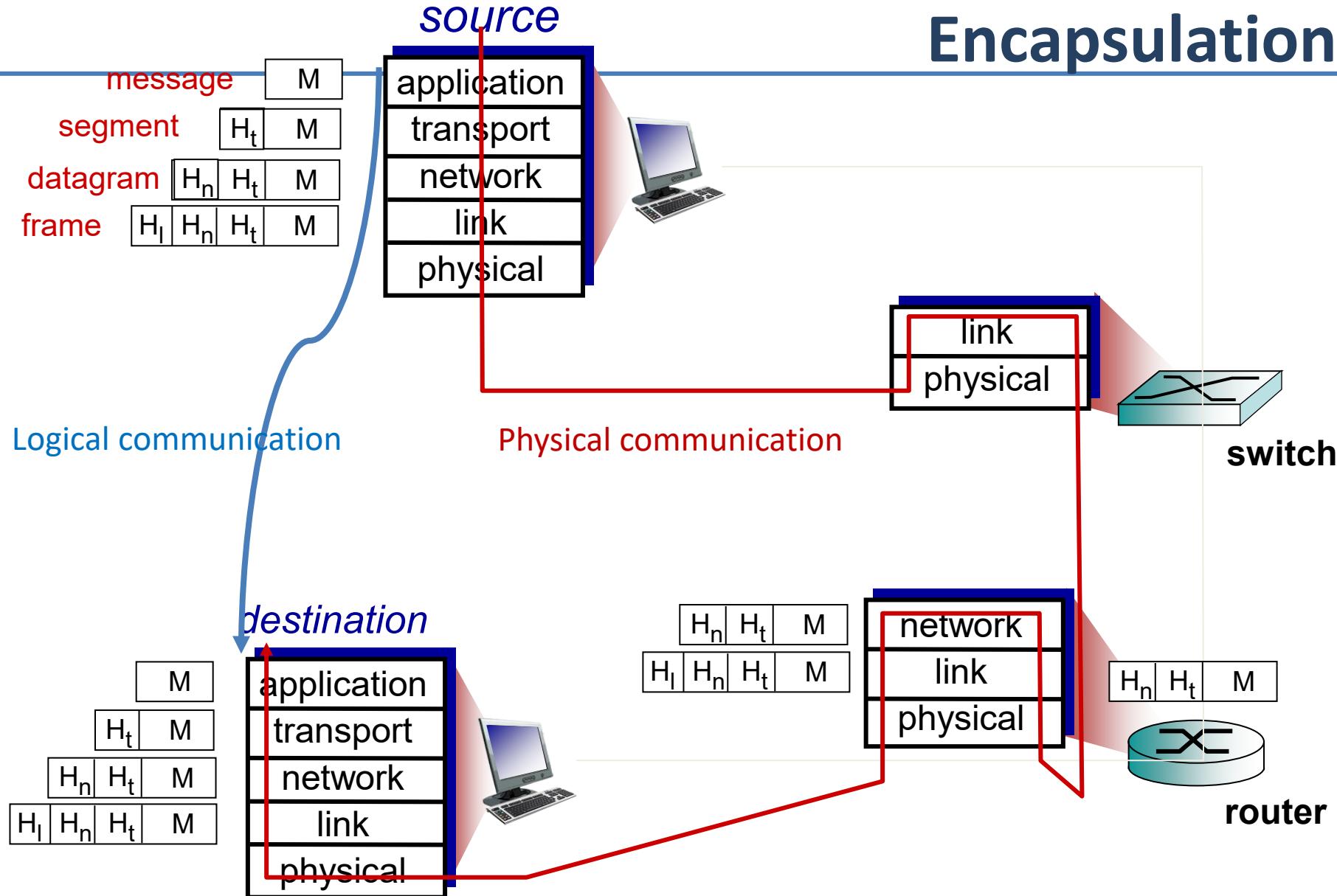


Roadmap

- What's the Internet
 - Nuts&bolts view
 - Service view
 - Distinction between network edge and network core
- Layers of abstraction, protocols
 - ISO/OSI & Internet layer structure
 - Data communication through layers: physical and logical view



Layered communication: Encapsulation



Chapter 1a: Summary

We discussed

- what's the Internet
- what's a protocol?
- protocol layers, service models

We will continue (next lecture) with

- Network edge & network core services & functionality overview
- More on Internet structure overview
 - access nets, physical media
 - backbones, NAPs, ISPs
- Performance concerns: delays, loss
- Some security concerns

To provide :

- context, overview, “feeling” of networking
- A point of reference for context in the more detailed discussions to come



Reading instructions (incl.next lecture)

Kurose Ross book

Careful

1.2--1.5

Quick

the rest

Extra Reading (optional)

Computer and Network Organization: An Introduction,
by Maarten van Steen and Henk Sips, Prentice Hall
(very good introductory book for students from programs other than CSE/IT)

Self-study review questions (incl. next lecture)

Review questions from Kurose-Ross book, chapter 1 (for basic study, incl. next lecture)

- R11, R12, R13, R16, 17, R18, R19, R20, R21, R22, R23, R24, R25, R28.



Course on Computer Communication and Networks

Lecture 2
Chapter 1: Introduction:
Part B: Internet structure; Data transfer performance; Security “prelude”

EDA344, DIT 423, LEU062

Based on the book Computer Networking: A Top Down Approach, Jim Kurose, Keith Ross, Addison-Wesley.

Recap – menti.com

Layering – Some history: The OSI Reference Model

ISO (International Standards Organization) defined the OSI (Open Systems Interconnect) model to help vendors create interoperable network implementation

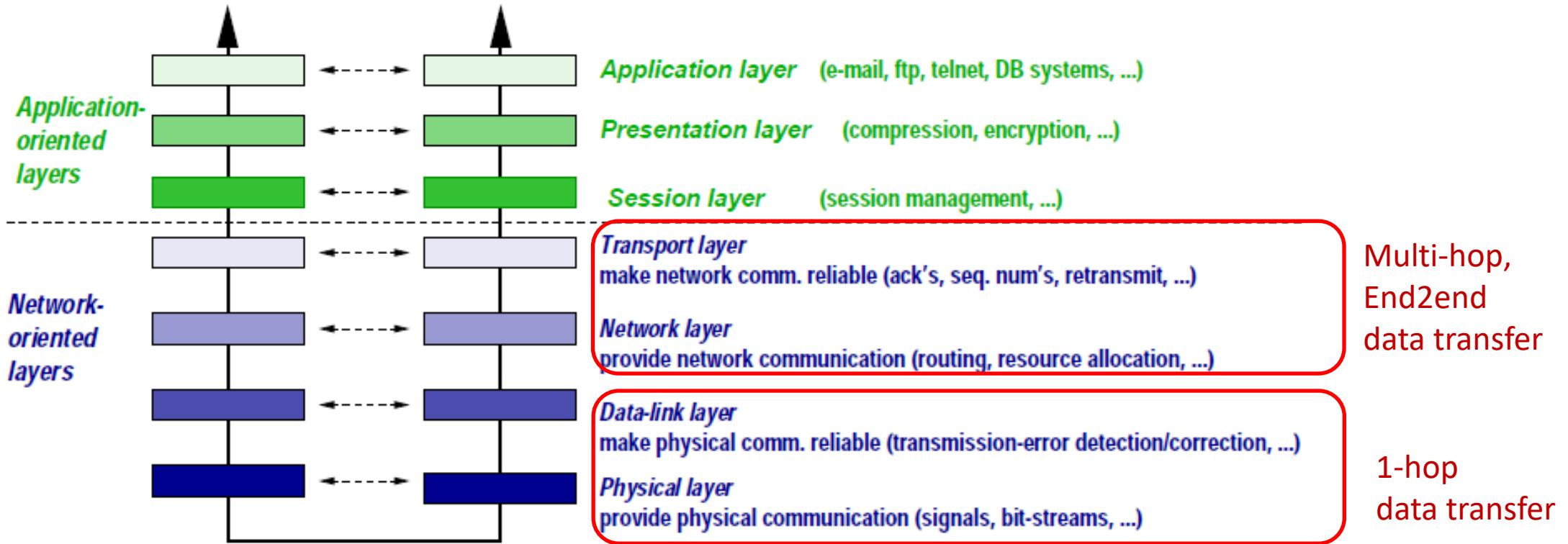


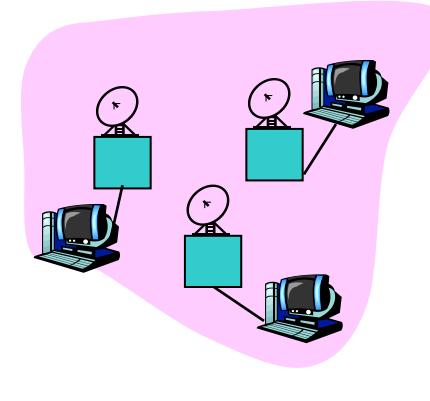
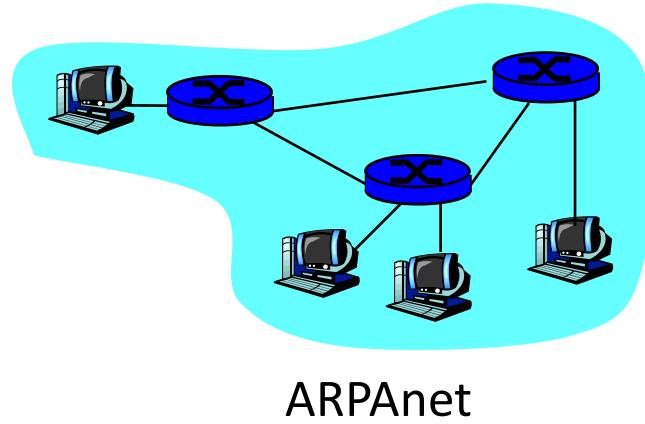
Fig. Steen, Sips : Computer and Network organization

``X dot'' series (X.25, X. 400, X.500) OSI model implementation (protocol stack)

The Internet concept: bridging networks

1974: multiple unconnected nets...

... differing in: addressing conventions, packet formats, routing, physical medium



satellite net

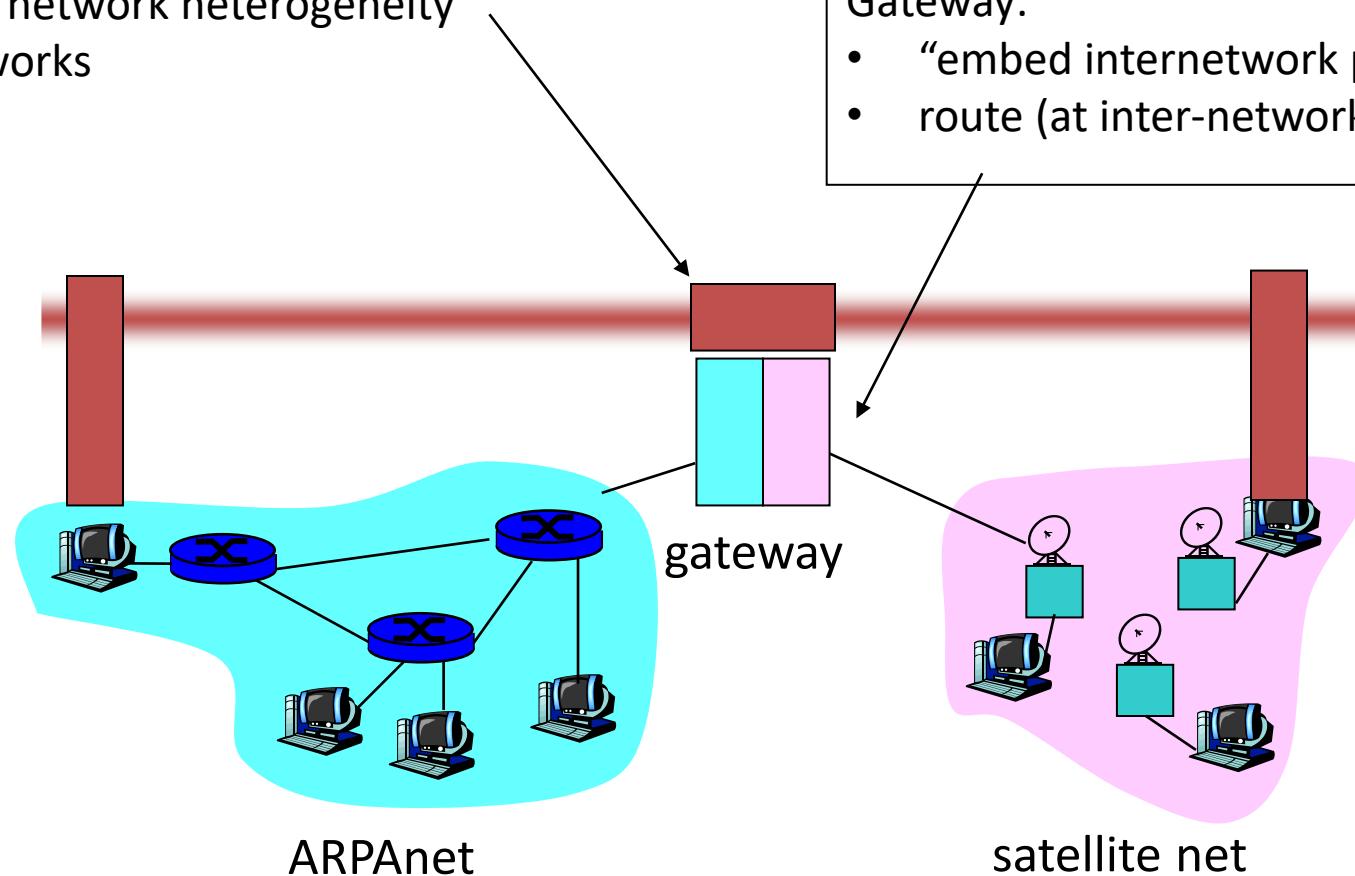
The Internet concept: bridging networks

Internet layer (IP):

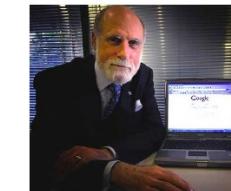
- internetwork appears as single, uniform entity, despite underlying local network heterogeneity
- network of networks

Gateway:

- “embed internetwork packets in local packet format”
- route (at inter-network level) to next gateway



1974



TCP / IP defined by Vint Cerf & Bob Kahn



2004: both received the Turing Award

"A Protocol for Packet Network Intercommunication", V. Cerf, R. Kahn, IEEE Transactions on Communications, May, 1974, pp. 637-648.

Internet protocol stack layers&protocols

Application: protocols supporting network applications

http (*web*), smtp (*email*), others (*p2p, streaming, ...*)

Transport: process2process/end2end data transfer/interface

UDP (user Datagram Prot.), TCP (Transmission Control Prot.), ...

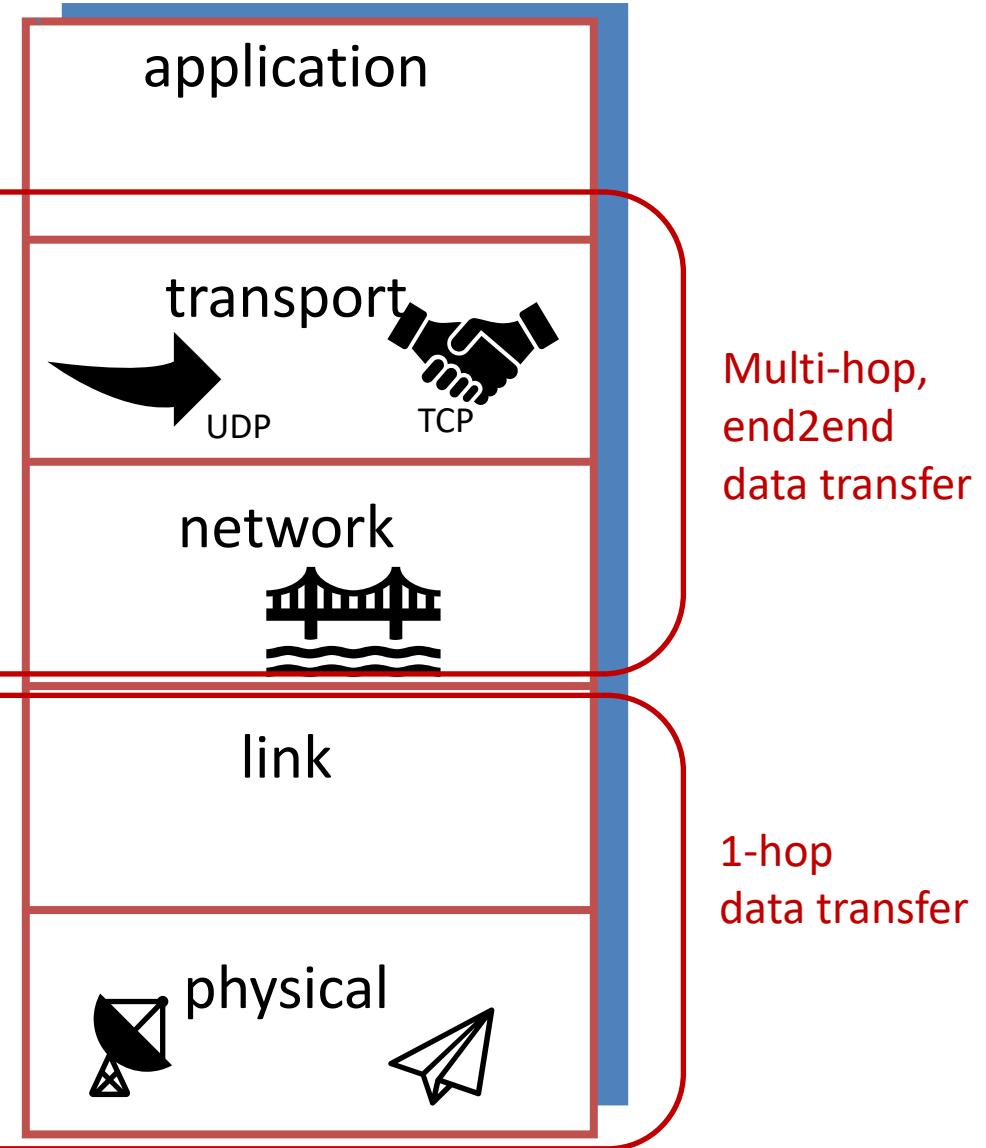
Network: routing of datagrams (independent data-packets), connecting nodes at different physical networks

Universal addressing to bridge NWs that have various addressing formats & routing protocols

Link: data transfer & synchronization between directly connected nodes

Ethernet, WiFi, ...

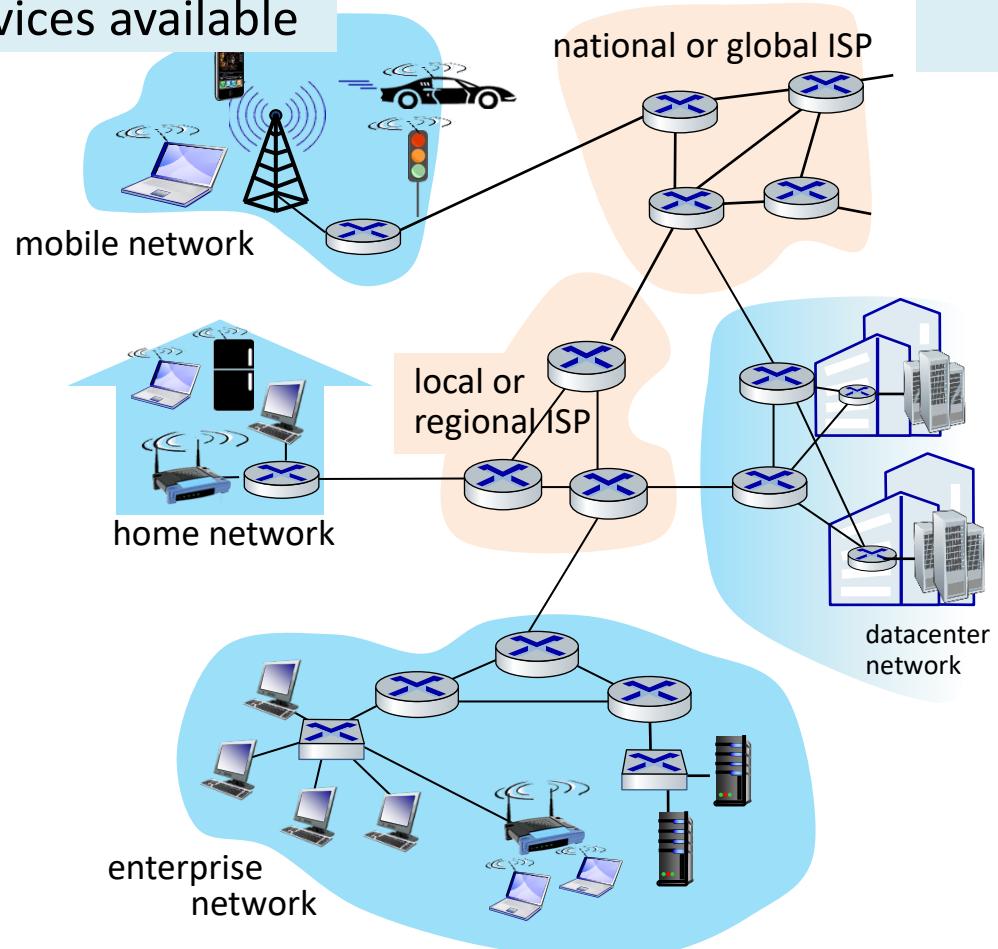
Physical: bit-transmission on the physical medium between directly connected nodes



(recall) A closer look at network structure:

network edge: end-hosts:

- run application programs e.g. Web, email, ...
- based on network services available



access networks /Inter-connection:

- connect end-hosts to the Internet
- through physical media: wired, wireless links

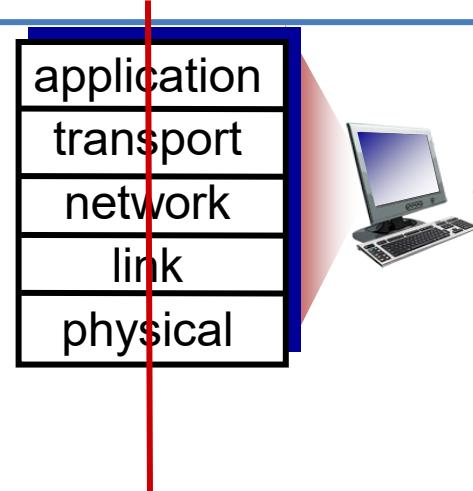
network core:

- interconnected routers
- network of networks

A closer look at network structure (cont)

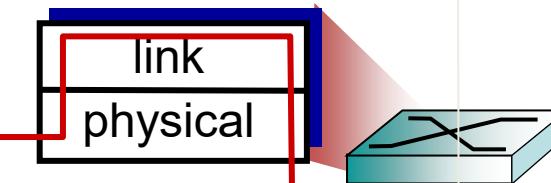
network edge: end-hosts:

- run application programs e.g. Web, email, ...
- based on network services available

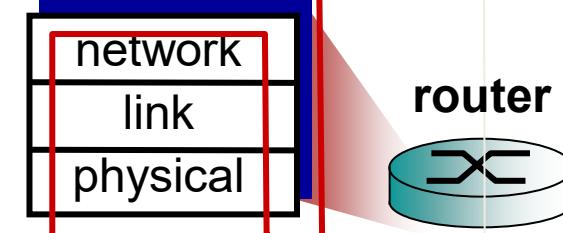
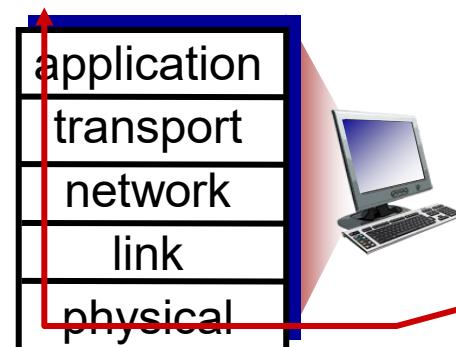


access networks /Inter-connection:

- connect end-hosts to the Internet
- through physical media: wired, wireless links



switch

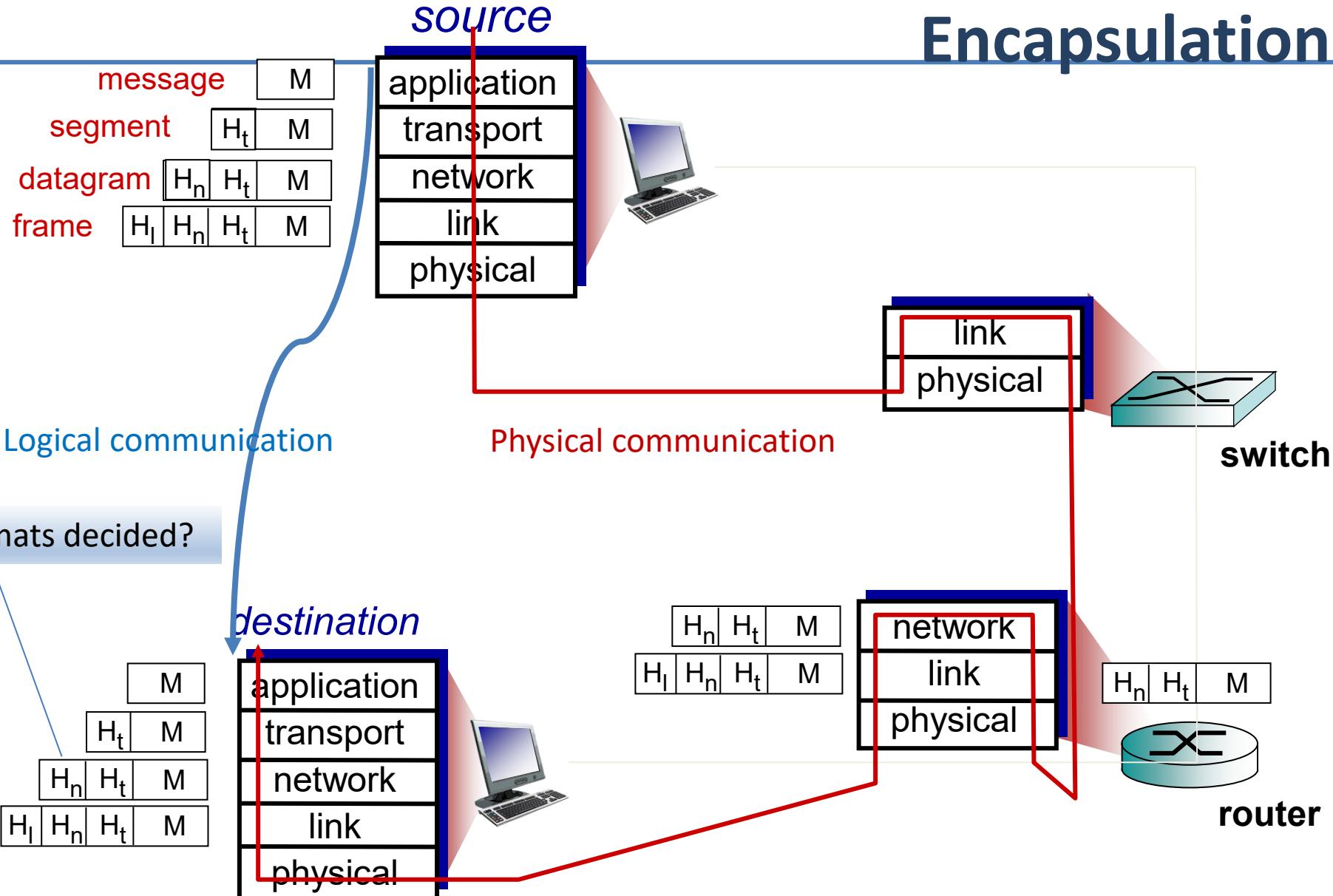


router

network core:

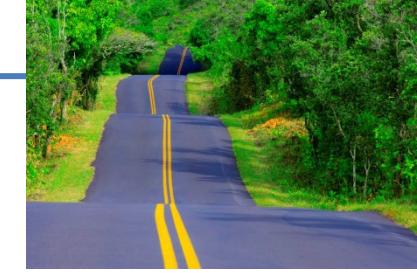
- interconnected routers
- network of networks

Layered communication: Encapsulation



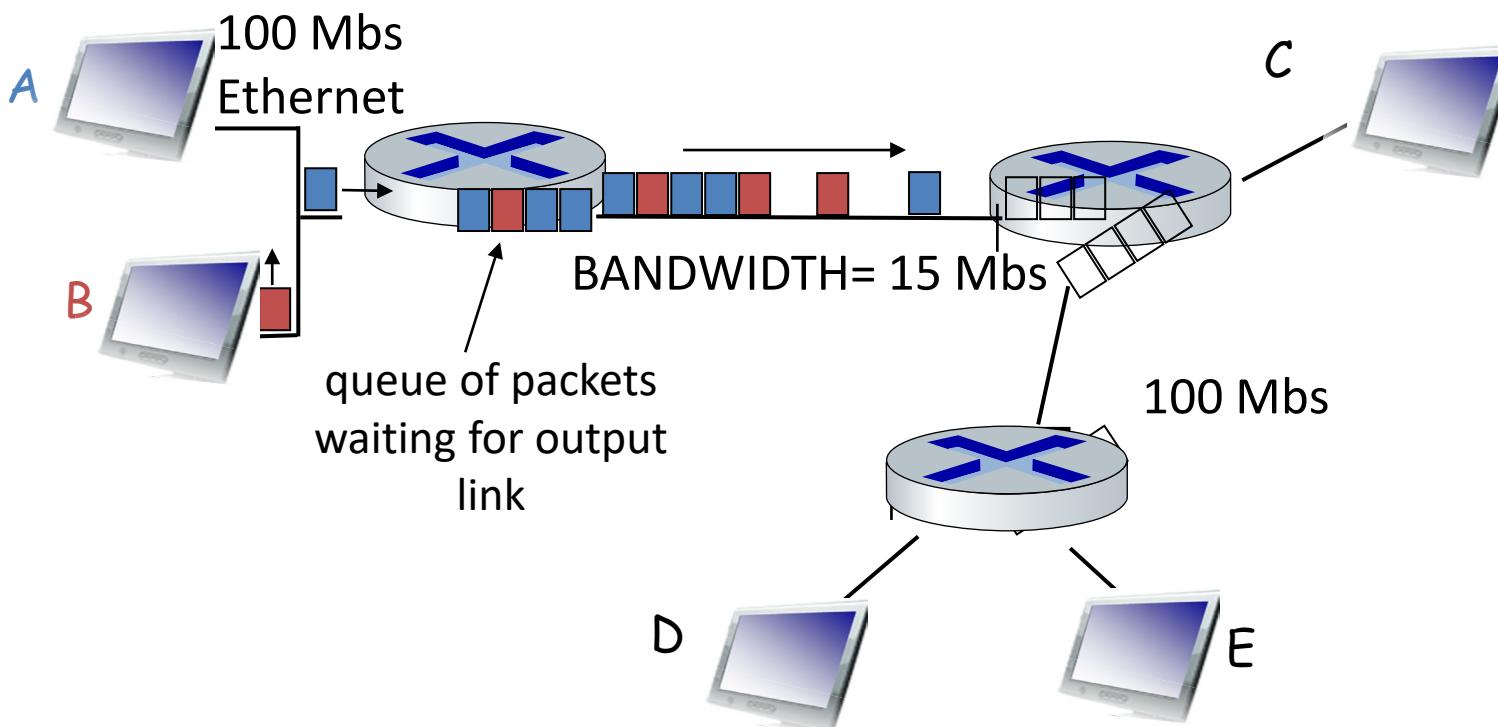
Roadmap

- 1. Zooming into **the NW core**
 - Data transfer through packet switching
 - Performance:
 - delays (aka latency) & loss
 - throughput
- 2. Inter-connection structure:
 - access net, physical media
 - backbones, NAPs, ISPs
- 3. Security prelude



Packet Switching

[part of L. Kleinrock's Contributions]



Application messages divided into *packets*

- packets *share* network resources

Store and forward:

- packets move one hop at a time
- transmit over link; wait their turn @next-link (queuing)

Delays (latency sources) in packet-switched networks

1. Transmission delay:

- R=link **bandwidth** (bps)
- L=packet length (bits)
- time for the “bits to hop onto line” = L/R

2. Propagation delay:

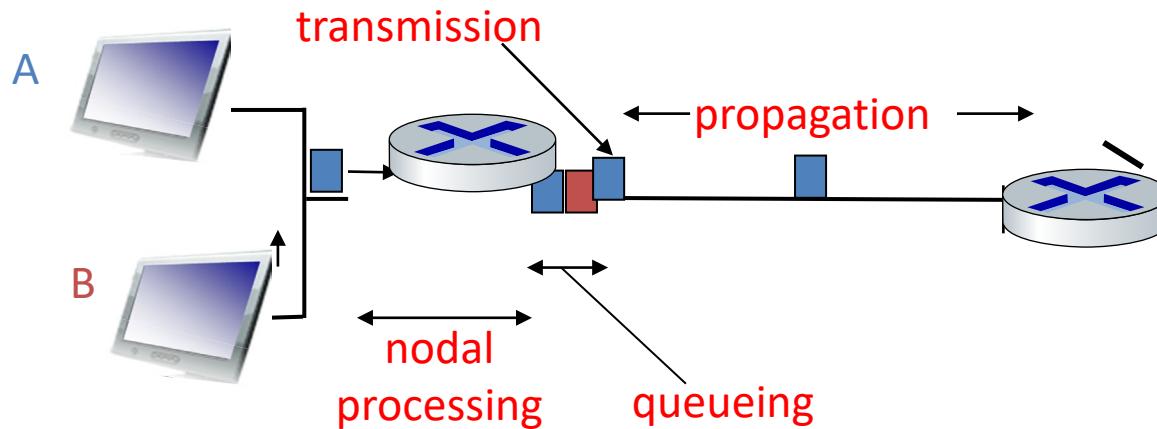
- d = **length** of physical link
- s = propagation speed in medium ($\sim 2 \times 10^8$ m/sec)
- Time for the bits to “travel on the line(s)” = d/s

3. nodal processing, e.g.:

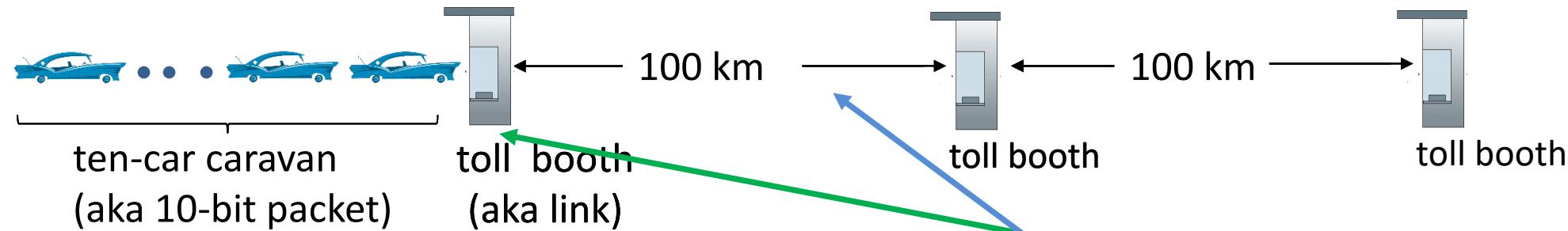
- check bit errors
- determine output link

4. queuing

- waiting at input link for processing, output link for transmission
- ~congestion @router



Transmission & propagation: caravan analogy



- car ~ bit; caravan ~ packet; toll service ~ link transmission
- toll booth takes 12 sec to service car (bit transmission time)
- “propagate” at 100 km/hr
- **Q:** How long until caravan is lined up before 2nd toll booth?
- **time to “push” (“transmit”) entire caravan through toll booth onto highway = $12 * 10 = 120$ sec**
- **time for last car to propagate from 1st to 2nd toll both: $100\text{km}/(100\text{km/hr}) = 1 \text{ hr}$**
- **A: 62 minutes**

Let's play a bit 😊

- https://media.pearsoncmg.com/ph/esm/ecs_kurose_compnetwork_8/cw
 - https://media.pearsoncmg.com/ph/esm/ecs_kurose_compnetwork_8/cw/content/interactiveanimations/transmission-vs-propogation-delay/transmission-propagation-delay-ch1/index.html

Visualize packet switching delays to calculate them!

Store&forward + pipelining behavior visualization

Transmission delay:
time to send L bits into link of capacity R bps
 $= L/R$ sec

Propagation delay:
Time to travel distance d with speed s m/sec
 $= d/s$ sec

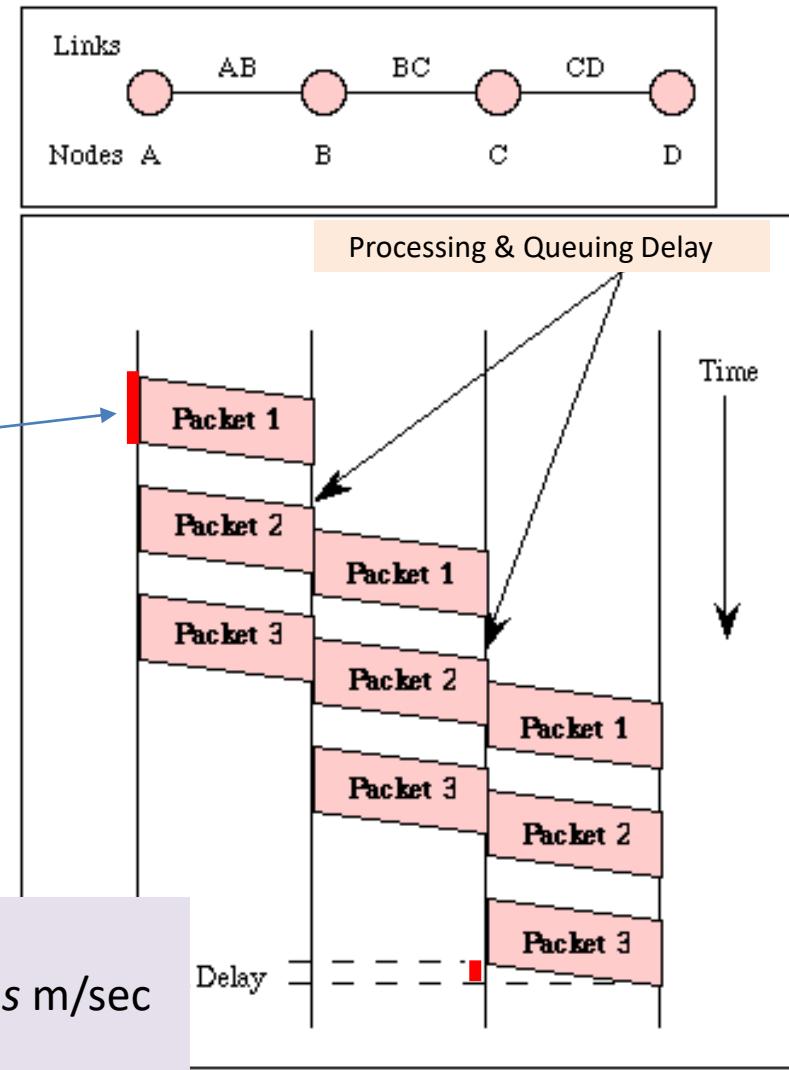


fig. Gorry Fairhurst

Some story-telling ...



Vinton G. Cerf

DOI:10.1145/3338516

Back to the Future, Part II

This year is the 50th anniversary of the activation of the Arpanet project. The first packet switches (called Interface Message Processors or IMPs) were installed at UCLA,

SRI International [then Stanford Research Institute], UC Santa Barbara, and University of Utah. Host computers at UCLA and SRI made their first connection through the Arpanet on October 29, 1969.

Tom Standage's book, *The Victorian Internet*,^a highlights the drama of the arrival of the telegraph in the mid-19th century. In its earliest incarnation, telegraph operators coded text

as well as other special characters. An operator would enter a message on the teletype and a paper tape would be punched. The operator would tear off the paper tape and hang it on a peg next to the teletype that would be used to send the message to the next hop. This was called a "torn tape" system since the punched paper tapes would be torn from the originating teletype and fed into a teletype that

even in the
microsecond
solution
switching
effective
introduced

"...the telephone emerged from research attempting to allow one wire to carry more than one telegraph message at the same time. ... Eventually **automatic [i.e. circuit] switching** was possible and **circuits were built automatically from source to destination**. This allowed teletypes to be directly connected end-to-end without requiring intermediate queueing. No more **store-and-forward [i.e. as in packet switching]** was necessary as the two end-points were directly connected by a circuit..."

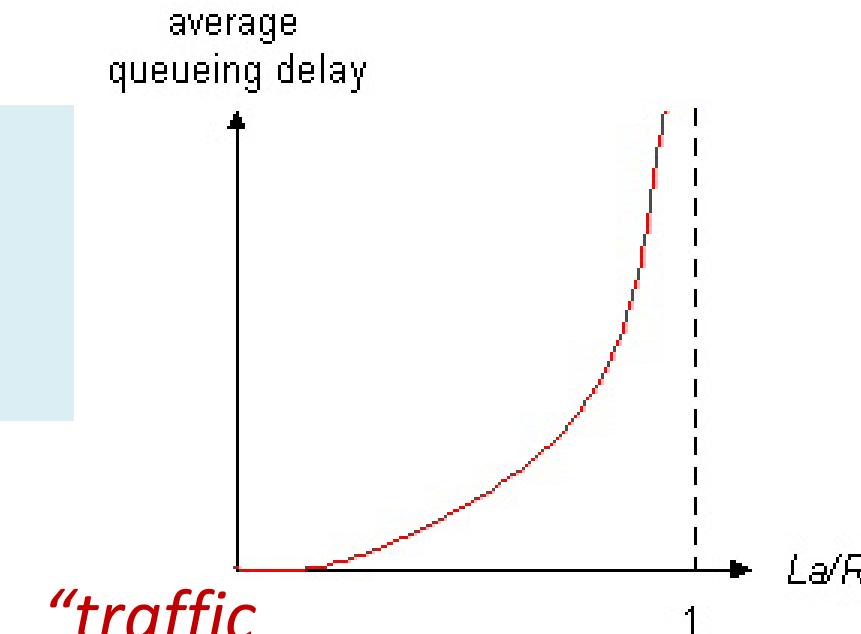
switching system! Since the circuits connecting the packet switches were fixed and dedicated, there was no circuit switching delay. Rather, packets from the host computers could

CACM 2019

Thinking about packet queueing delay ...

- R=link bandwidth (bps)
- L=packet length (bits)
- a=average packet arrival rate

$$\frac{L \cdot a}{R} : \frac{\text{arrival rate of bits}}{\text{service rate of bits}}$$



*"traffic
intensity"*

- $La/R \sim 0$: average queueing delay small
- $La/R \rightarrow 1$: queues become large!
- $La/R > 1$: more "work" arriving than can be serviced - Queues grow unlimited, unstable/saturated system



$La/R \sim 0$



La/R is high

Cycle Time vs. Throughput for Different Variability Levels

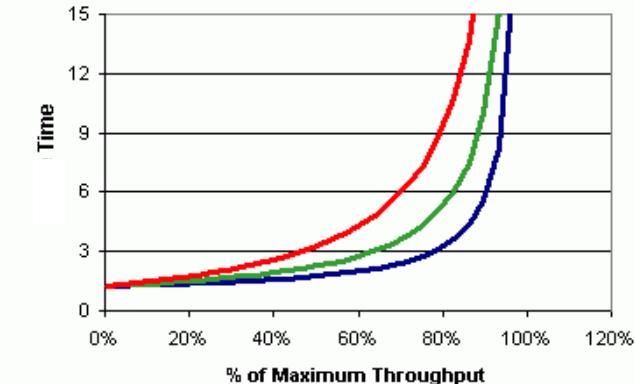
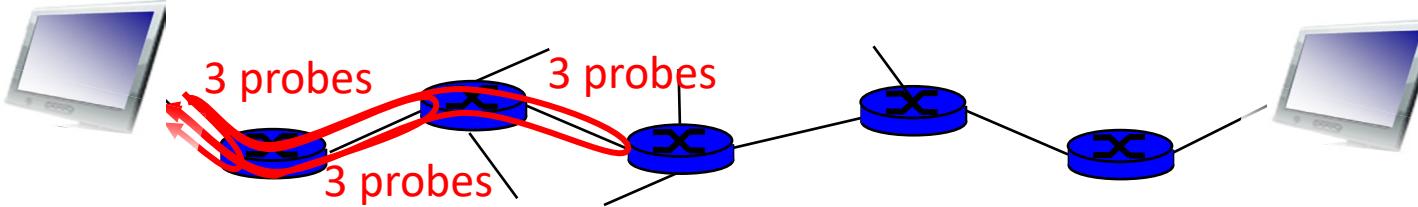


Fig. M. Lapedus, Semiconductor Eng. 2017

Curve aka "the queuing knee"

... “Real” Internet delays and routes (1)...

- What do “real” Internet delay & loss look like?
- Traceroute program: provides delay measurements from source to each router on end-end Internet path towards destination. For all i :
 - sends 3 pkts to router i on path towards destination
 - router i will return packets to sender
 - sender times the interval between transmission and reply.



...“Real” Internet delays and routes (2)...

traceroute: gaia.cs.umass.edu to www.eurecom.fr

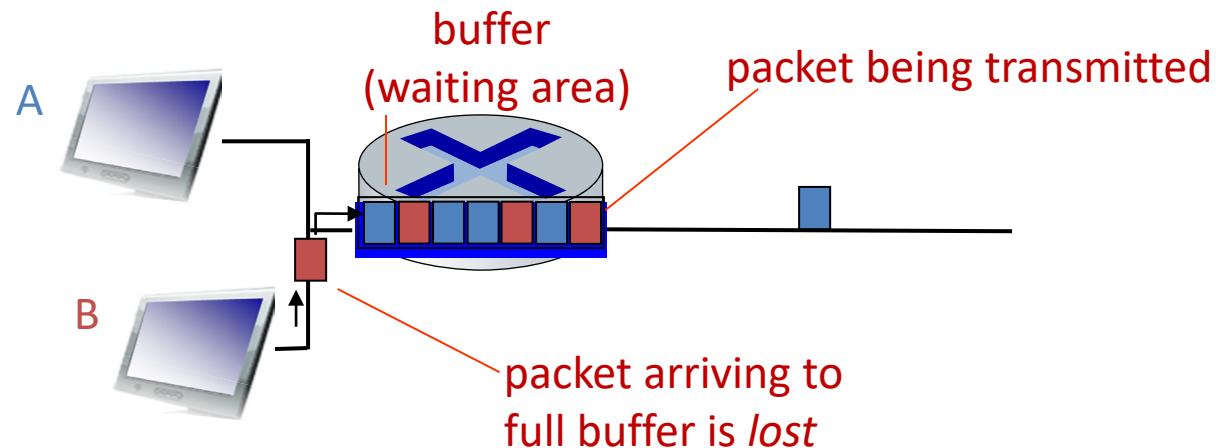
Three delay measurements from gaia.cs.umass.edu to cs-gw.cs.umass.edu						
1	cs-gw (128.119.240.254)	1 ms	1 ms	2 ms		
2	border1-rt-fa5-1-0.gw.umass.edu (128.119.3.145)	1 ms	1 ms	2 ms		
3	cht-vbns.gw.umass.edu (128.119.3.130)	6 ms	5 ms	5 ms		
4	jn1-at1-0-0-19.wor.vbns.net (204.147.132.129)	16 ms	11 ms	13 ms		
	jn1-so7-0-0-0.wae.vbns.net (204.147.136.136)	21 ms	18 ms	18 ms		
	abilene-vbns.abilene.ucaid.edu (198.32.11.9)	22 ms	18 ms	22 ms		
	nycm-wash.abilene.ucaid.edu (198.32.8.46)	22 ms	22 ms	22 ms		
8	62.40.103.253 (62.40.103.253)	104 ms	109 ms	106 ms		
9	de2-1.de1.de.geant.net (62.40.96.129)	109 ms	102 ms	104 ms		
10	de.fr1.fr.geant.net (62.40.96.50)	113 ms	121 ms	114 ms		
11	renater-gw.fr1.fr.geant.net (62.40.103.54)	112 ms	114 ms	112 ms		
12	nio-n2.cssi.renater.fr (193.51.206.13)	111 ms	114 ms	116 ms		
13	nice.cssi.renater.fr (195.220.98.102)	123 ms	125 ms	124 ms		
14	r3t2-nice.cssi.renater.fr (195.220.98.110)	126 ms	126 ms	124 ms		
15	eurecom-valbonne.r3t2.ft.net (193.48.50.54)	135 ms	128 ms	133 ms		
16	194.214.211.25 (194.214.211.25)	126 ms	128 ms	126 ms		
17	***					
18	***				* means no response (probe lost, router not replying)	
19	fantasia.eurecom.fr (193.55.113.142)	132 ms	128 ms	136 ms		

Q: Can some line have
eg 2 small values
& 1 big one?

Q: how can a packet get lost?

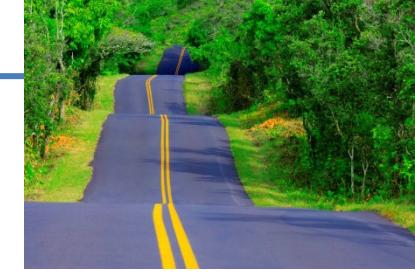
Delays and packet loss

- Link queue (aka buffer) has finite capacity
- packet arriving to full queue can cause **packet drops** (aka **loss**)



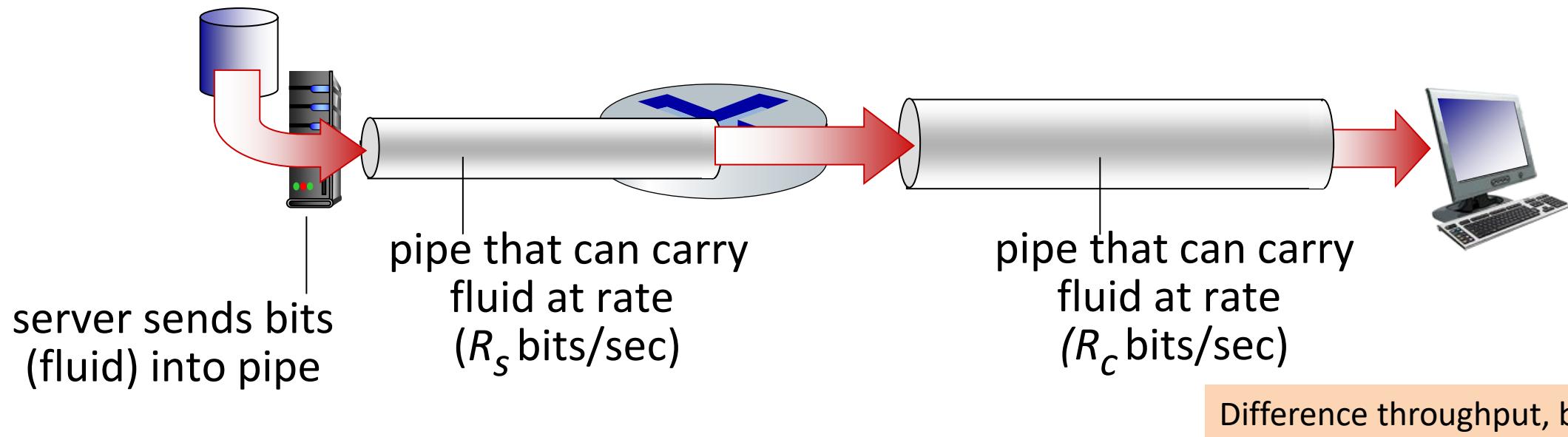
Roadmap

1. Zooming into **the NW core**
 - Data transfer through packet switching
 - Performance:
 - delays (aka latency) & loss
 - throughput
2. Inter-connection structure:
 - access net, physical media
 - backbones, NAPs, ISPs
3. Security prelude



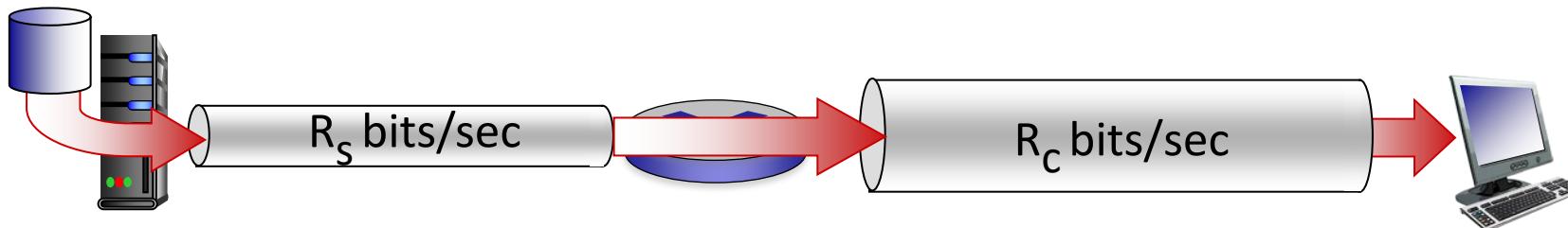
Throughput

- **throughput:** rate (bits/time unit) at which bits are being sent from sender to receiver
 - *instantaneous:* rate at given point in time
 - *average:* rate over longer period of time

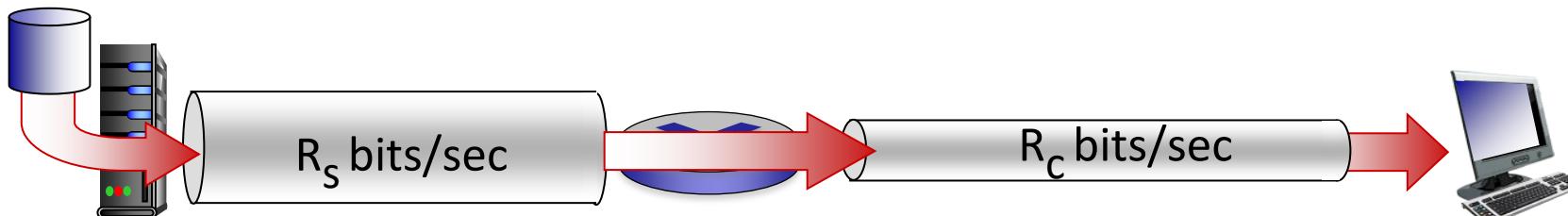


Throughput

$R_s < R_c$ What is average end-end throughput?



$R_s > R_c$ What is average end-end throughput?



bottleneck link

Smallest-bandwidth (aka capacity) transmission link on end-end path
(it constrains end-end throughput)

Roadmap

1. Zooming into **the NW core**
 - Data transfer through packet switching
 - Performance:
 - delays (aka latency) & loss
 - throughput
2. Inter-connection structure:
 - access net, physical media
 - backbones, NAPs, ISPs
3. Security prelude



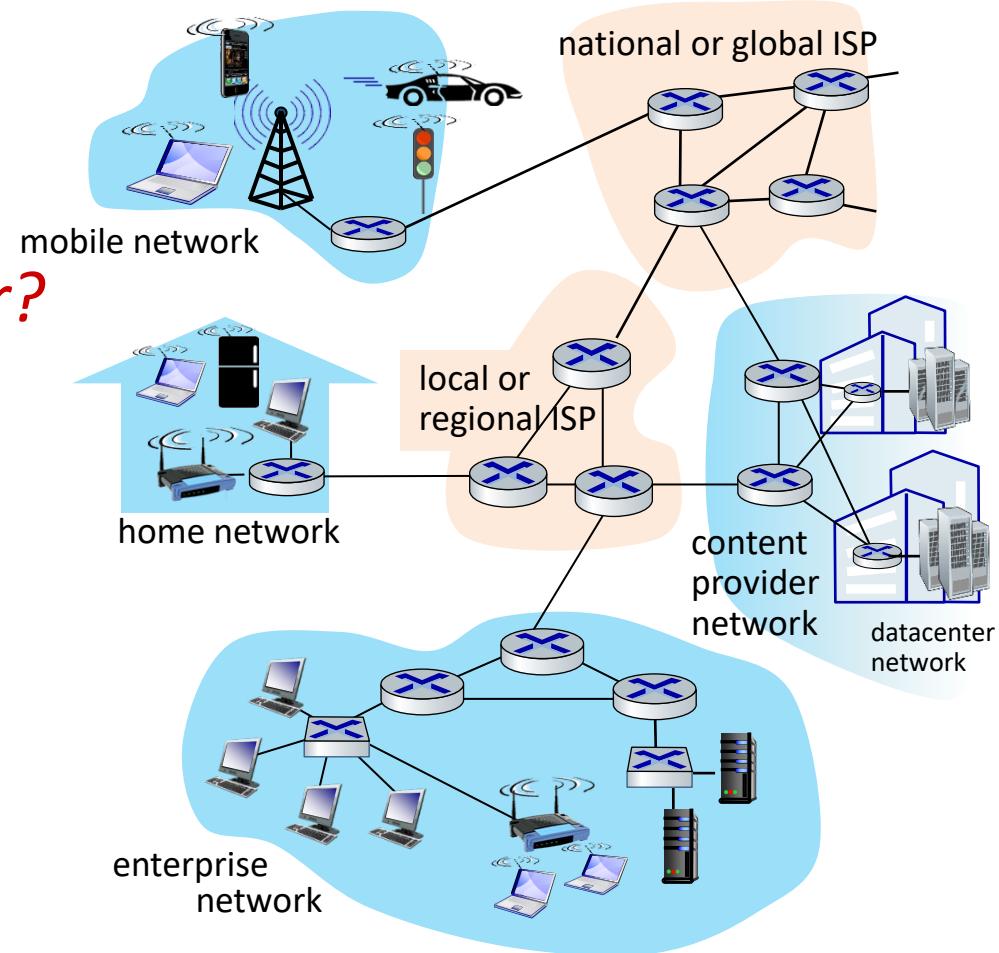
Access networks and physical media

Q: How to connect end systems to first router?

- Media: cable, wireless
- access nets: residential, institutional (school, company)

keep in mind:

- bandwidth (bits per second)?
- shared or dedicated?

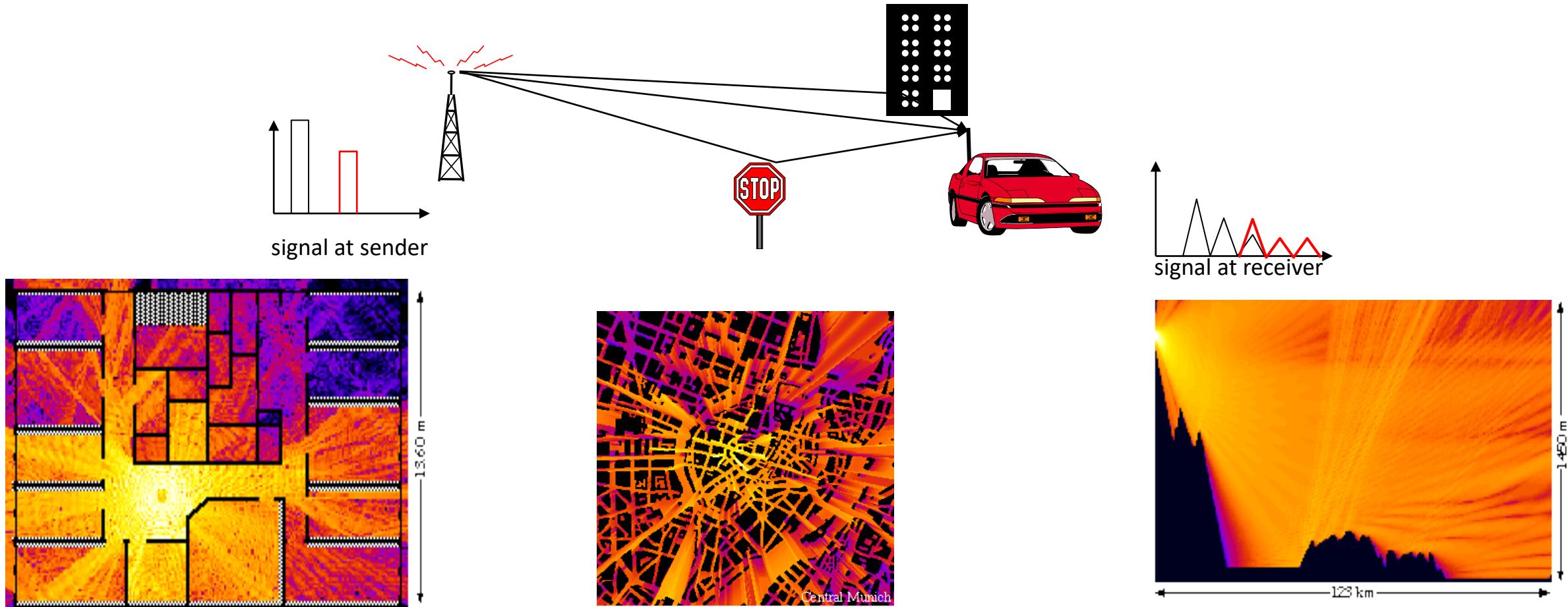


Unguided Physical media: Radio

Watch for: Attenuation, Multipath propagation, Interference

Radio links (Mbps): terrestrial microwave, LAN/WiFi, wide-area/cellular, satellite

Signal can **fade** with distance or get **obstructed** or get **reflected** and take many different paths



Guided Physical Media: coax, fiber, twisted pair

Coaxial cable:

- wire (signal carrier) within a wire (shield)
- broadband: multiple channels multiplexed on cable (HFC, cable TV)
- 100's Mbps per channel



Fiber optic cable:

- low attenuation: hence fewer repeaters
- low error rate: light pulses immune to electromagnetic noise
- high-speed operation: e.g., 10-100 Gbps



Twisted Pair (TP)

- two insulated copper wires
 - Category 5: 100 Mbps – 1Gbps Ethernet
 - Category 5/6: more twists, higher insulation: 10 Gbps Ethernet



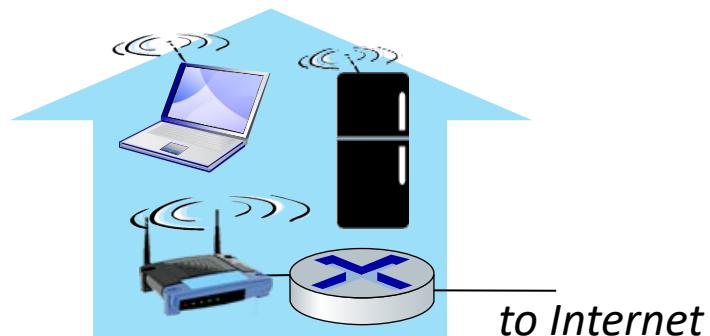
Wireless access networks

Shared *wireless* access network connects end system to router

- via base station aka “access point”

Wireless local area networks (WLANs)

- typically within or around building (~10-100m)
- 802.11b/g/n (WiFi): 11, 54, 450 Mbps transmission rate



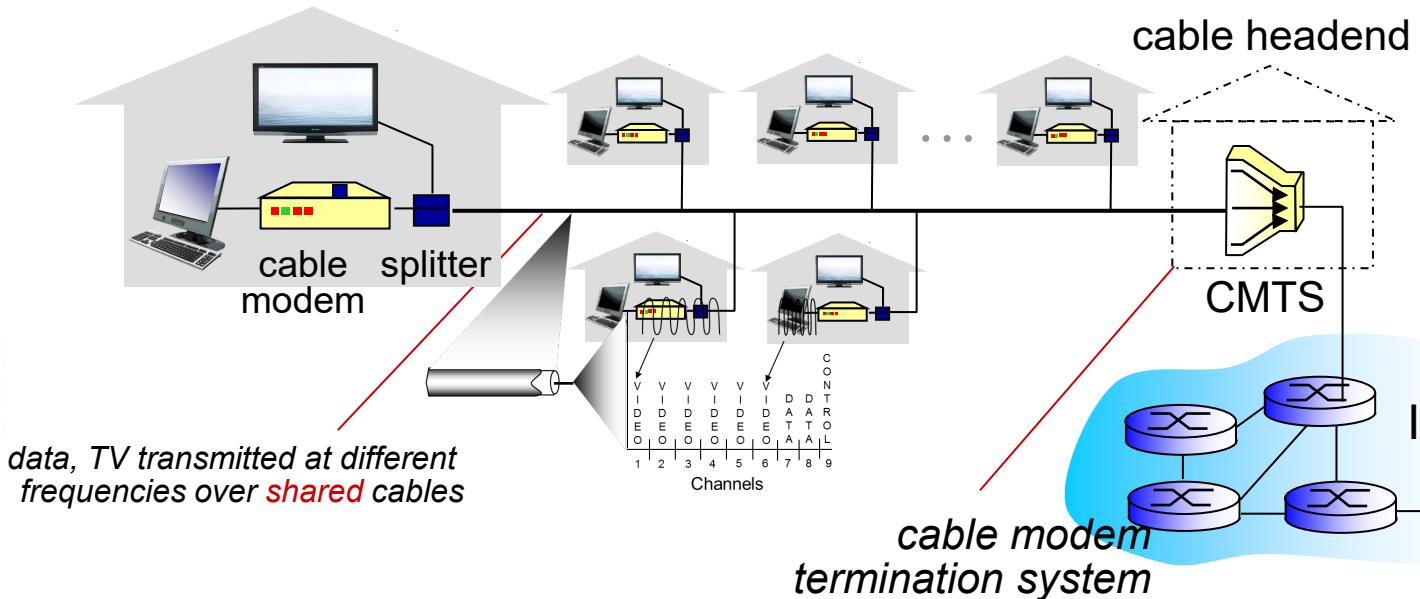
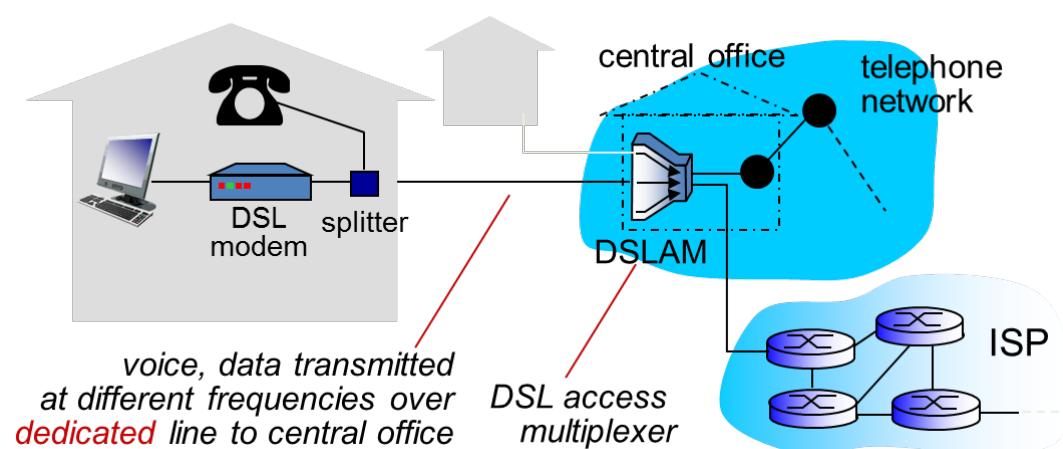
Wide-area cellular access networks

- provided by mobile, cellular network operator (10's km)
- 10+'s Mbps
- 4G/5G cellular networks (evolving)



Access nets examples:

digital subscriber line (DSL) & cable network (HFC)



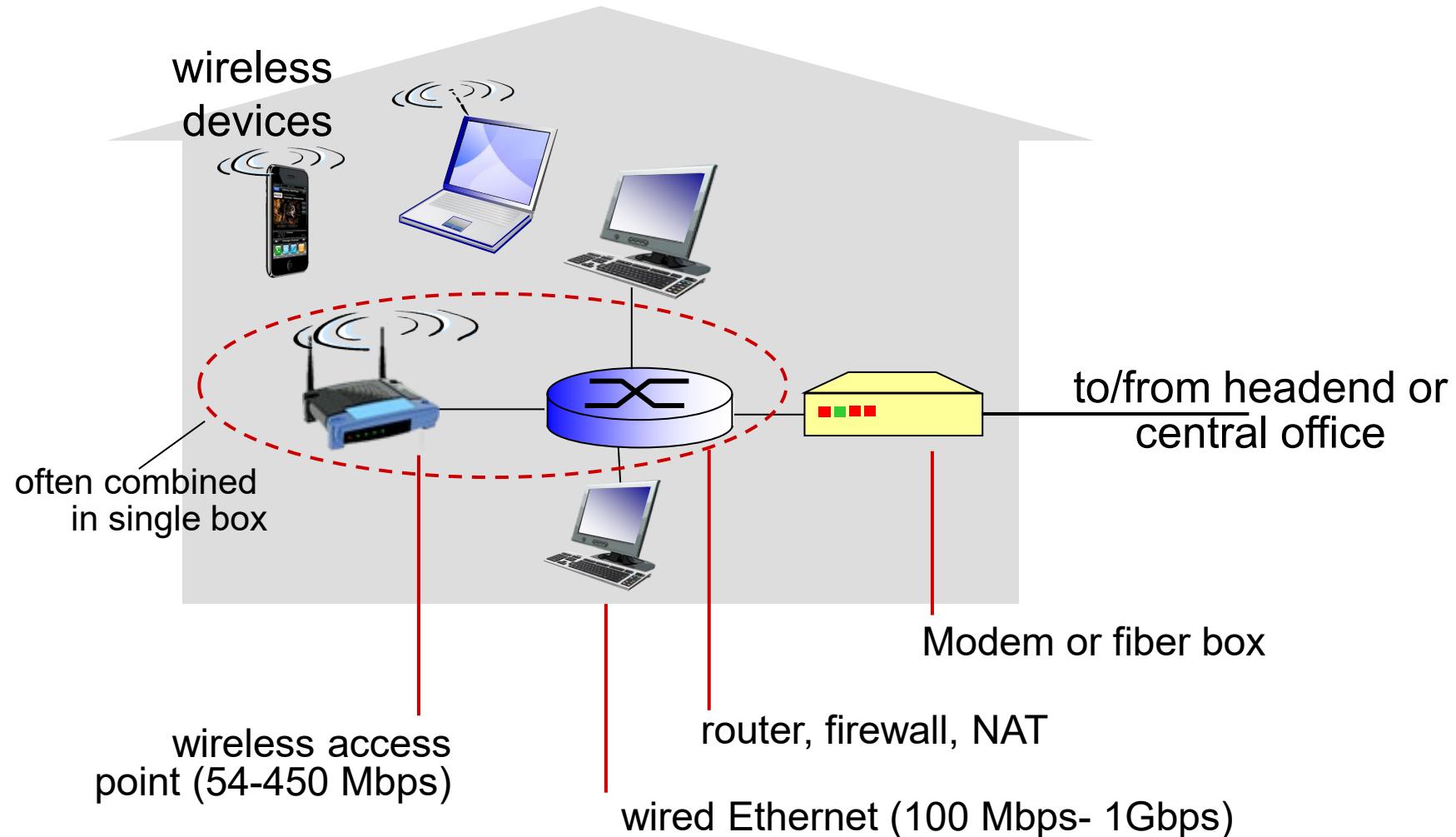
DSL: use *existing dedicated* phone line to central office

- multiplexes data/voice over DSL phone line to Internet/telephone net
- typically 3-16 Mbps upstream, ~20-50 Mbps downstream

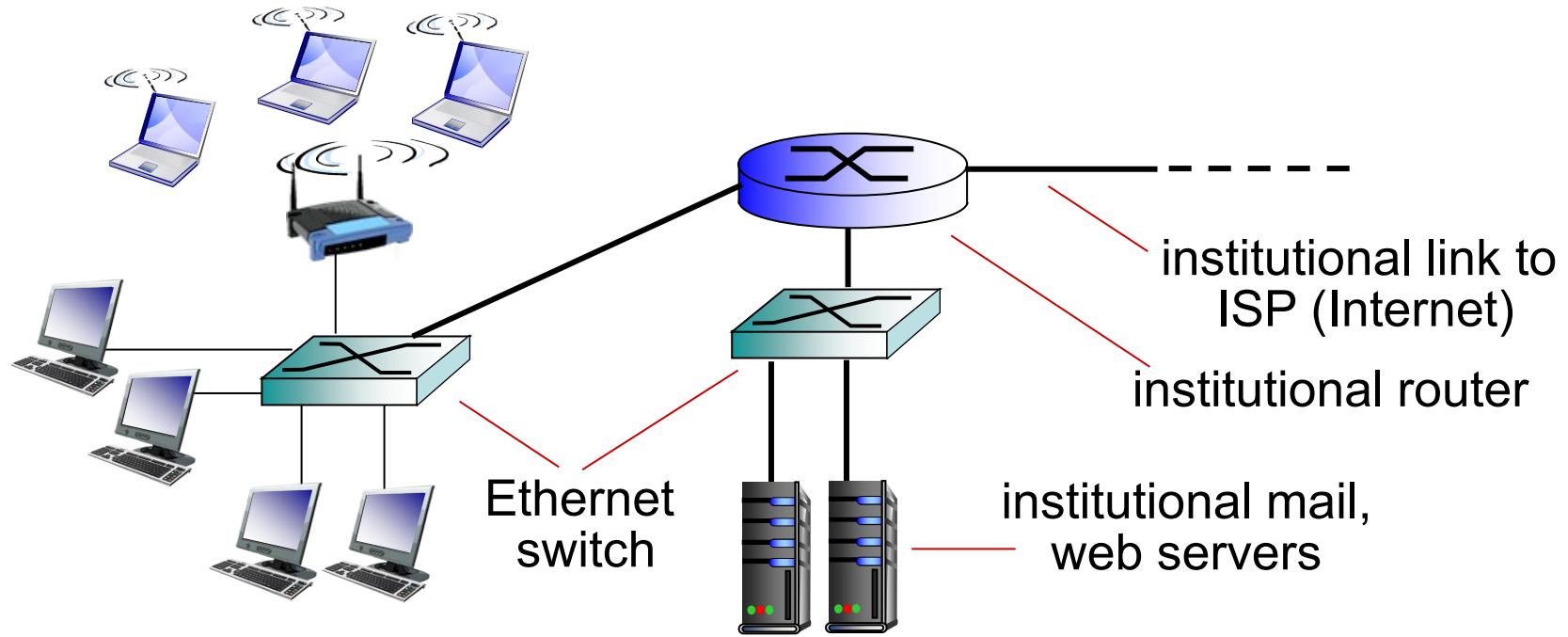
HFC (hybrid fiber coax): attach homes to ISP router

- 30-100 Mbps upstream, ~1.2Gbps downstream,
- *shared access network* to cable headend

Access net: home network



Enterprise access networks (Ethernet)



Typically used in companies, universities, etc

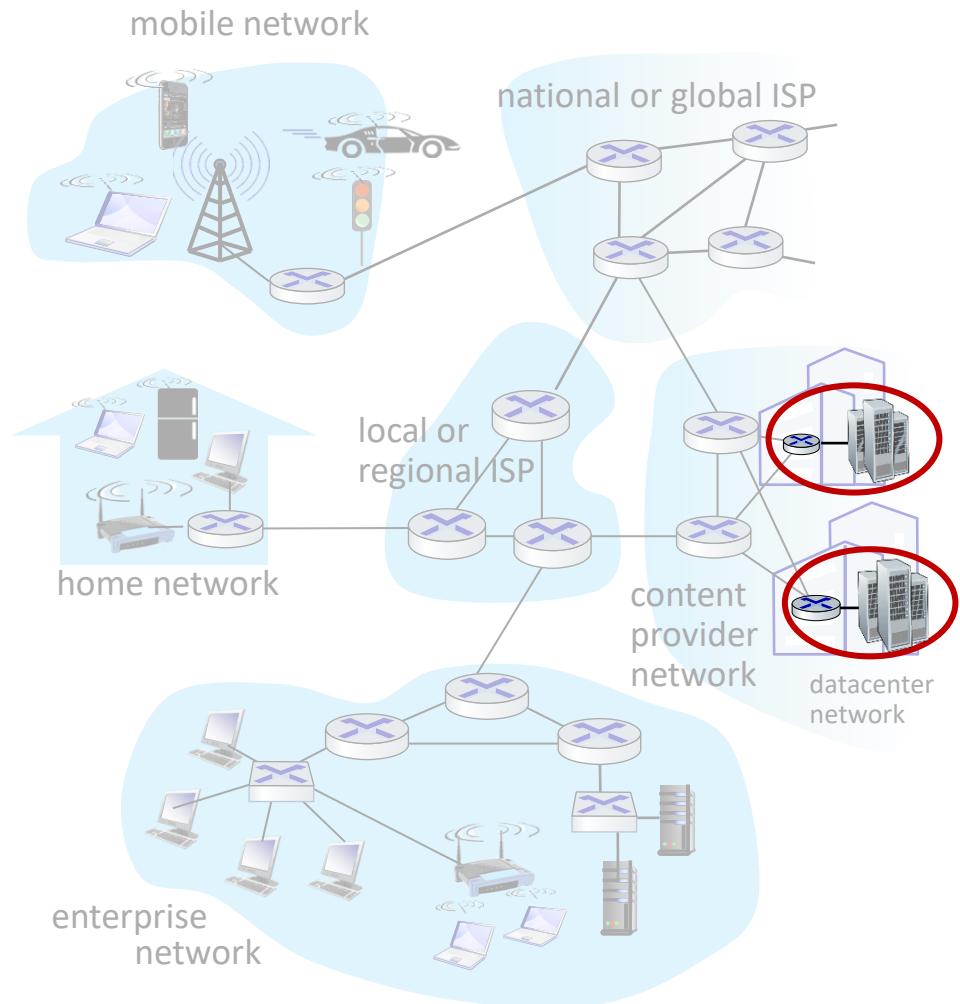
- Ethernet: wired access at 100Mbps, 1Gbps, 10Gbps
- WiFi: wireless access points at 11, 54, 450 Mbps

Access networks: data center networks

- high-bandwidth links (10s to 100s Gbps) connect hundreds to thousands of servers together, and to Internet



Courtesy: Massachusetts Green High Performance Computing Center (mghpcc.org)



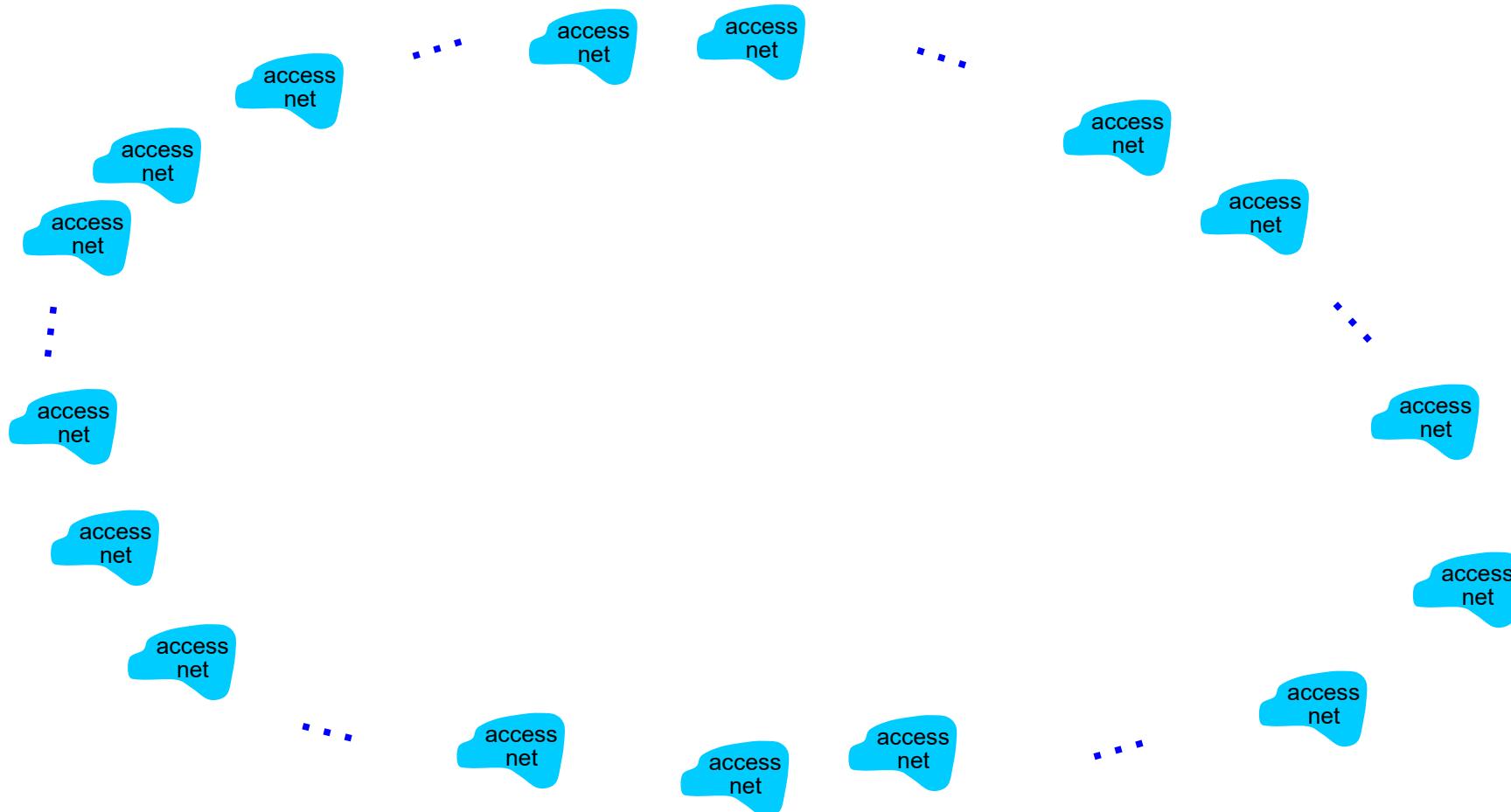
Roadmap



1. Zooming into **the NW core**
 - Data transfer through packet switching
 - Performance: delays (aka latency) & loss
 - throughput
2. Inter-connection structure:
 - access net, physical media
 - backbones, NAPs, ISPs
3. Security prelude

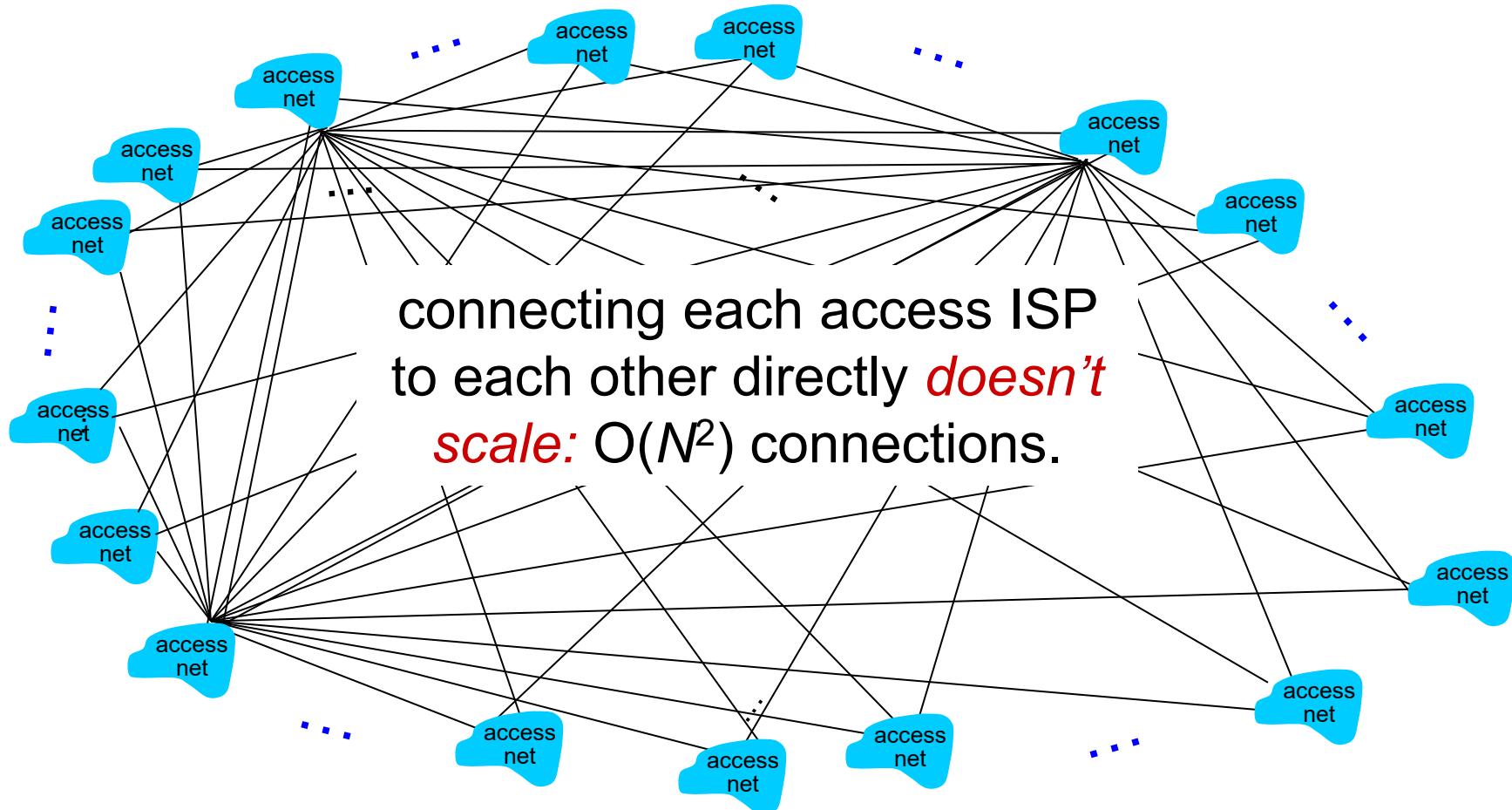
Internet structure: network of networks

Question: given *millions* of access ISPs, how to connect them together?



Internet structure: network of networks

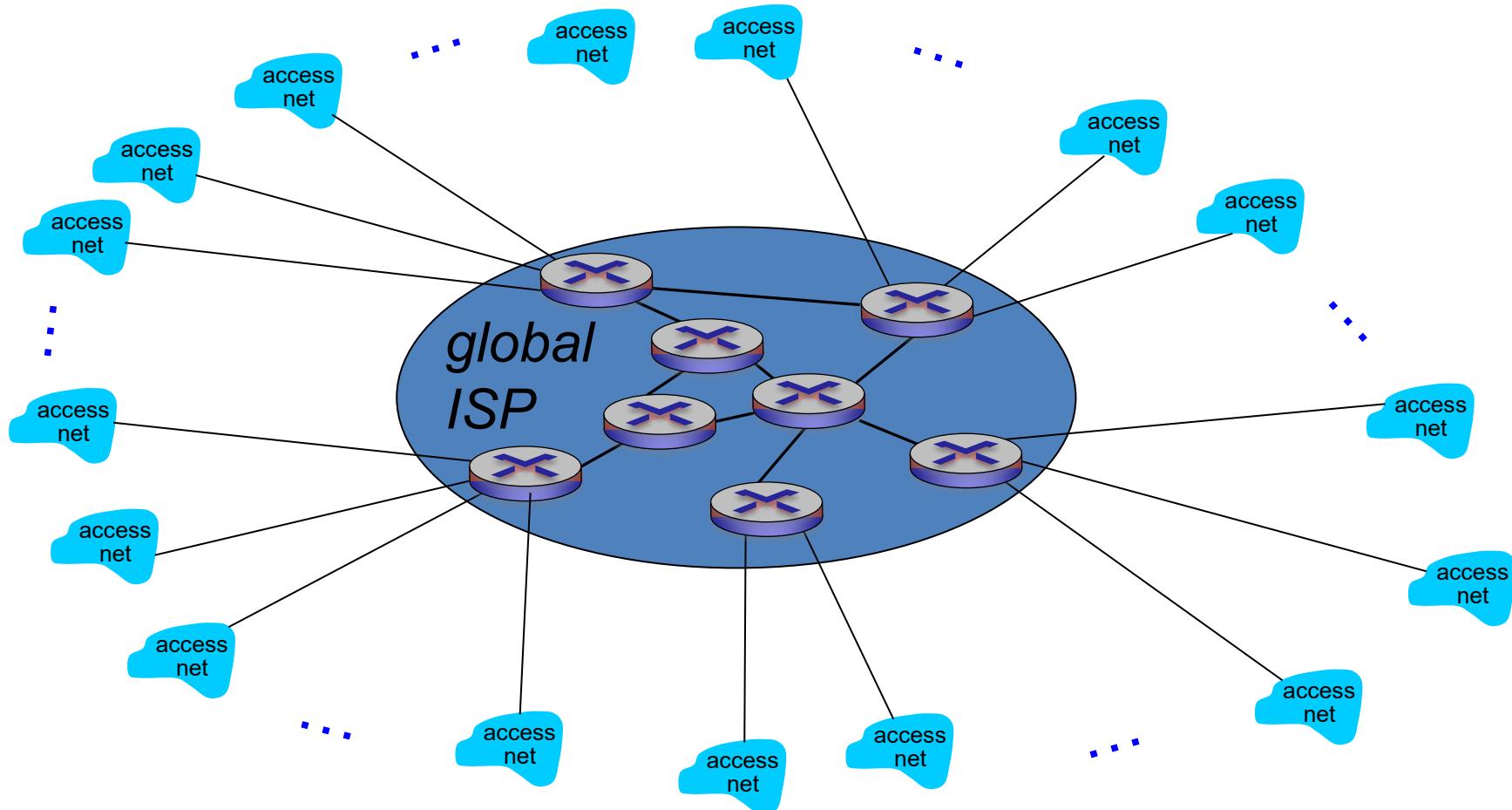
Option: connect each access ISP to every other access ISP?



Internet structure: network of networks

Option: connect each access ISP to one *global transit ISP*?

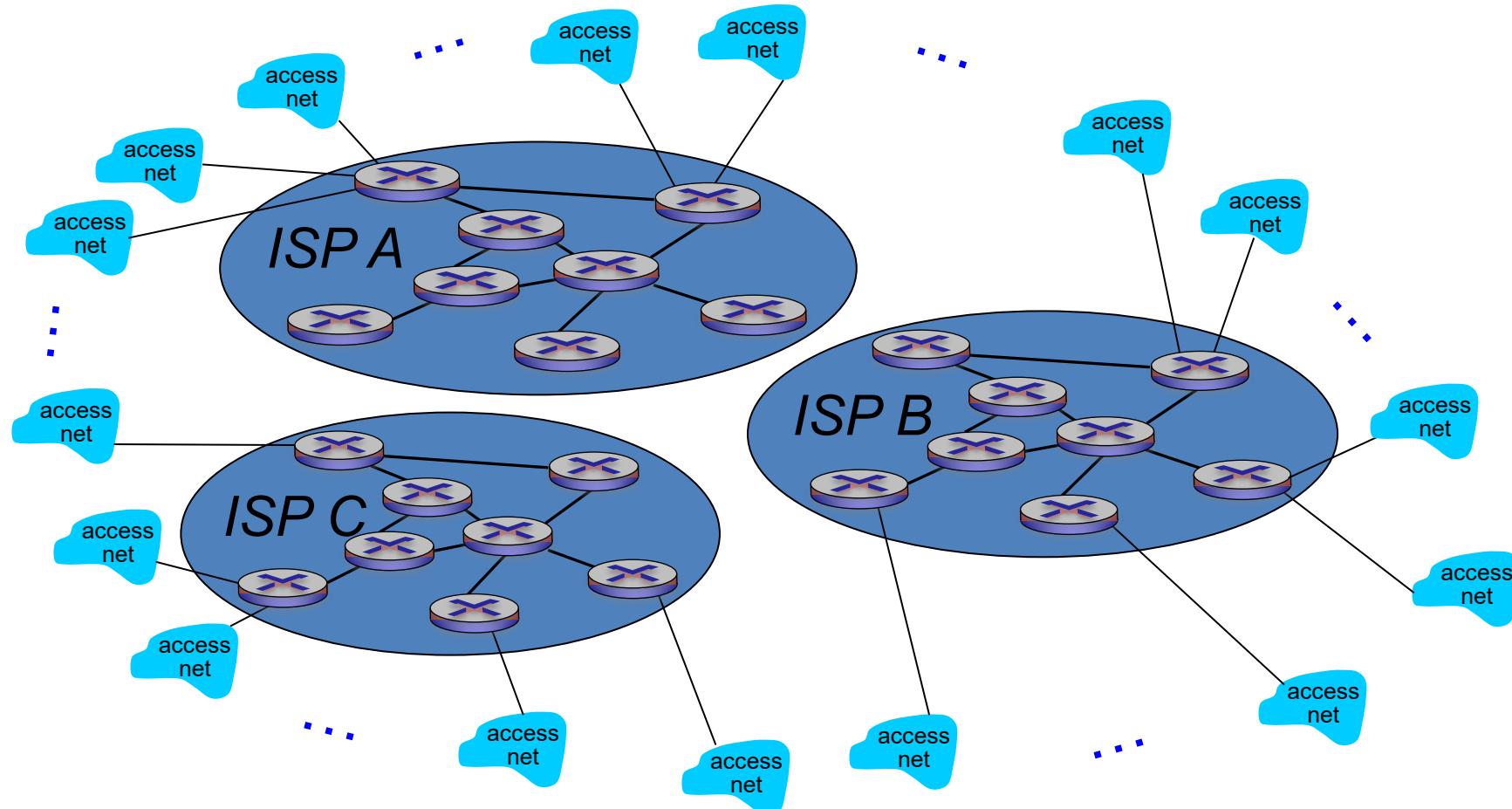
Customer and provider ISPs have economic agreement.



Internet structure: network of networks

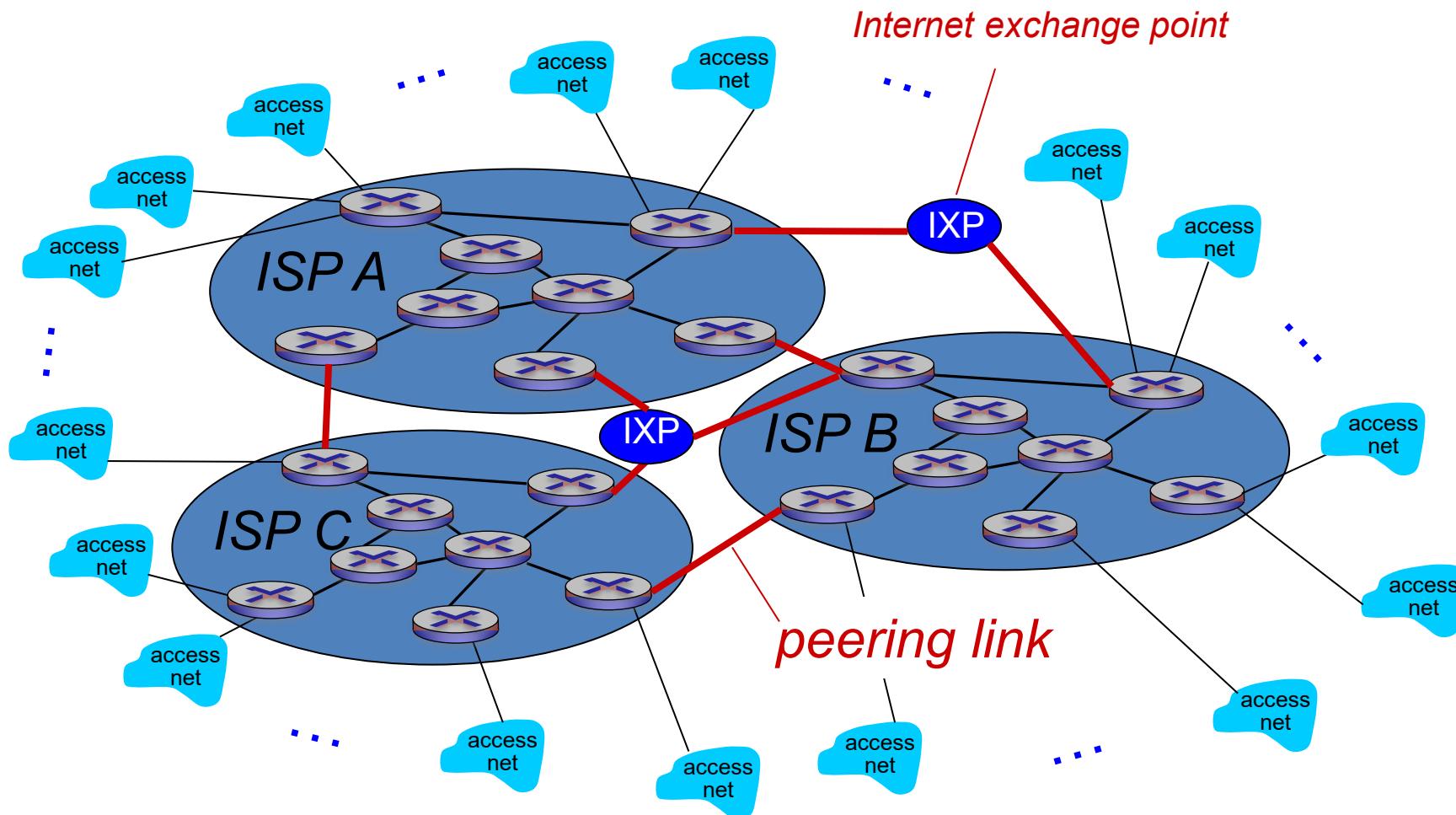
But if one global ISP is viable business, there will be competitors

....



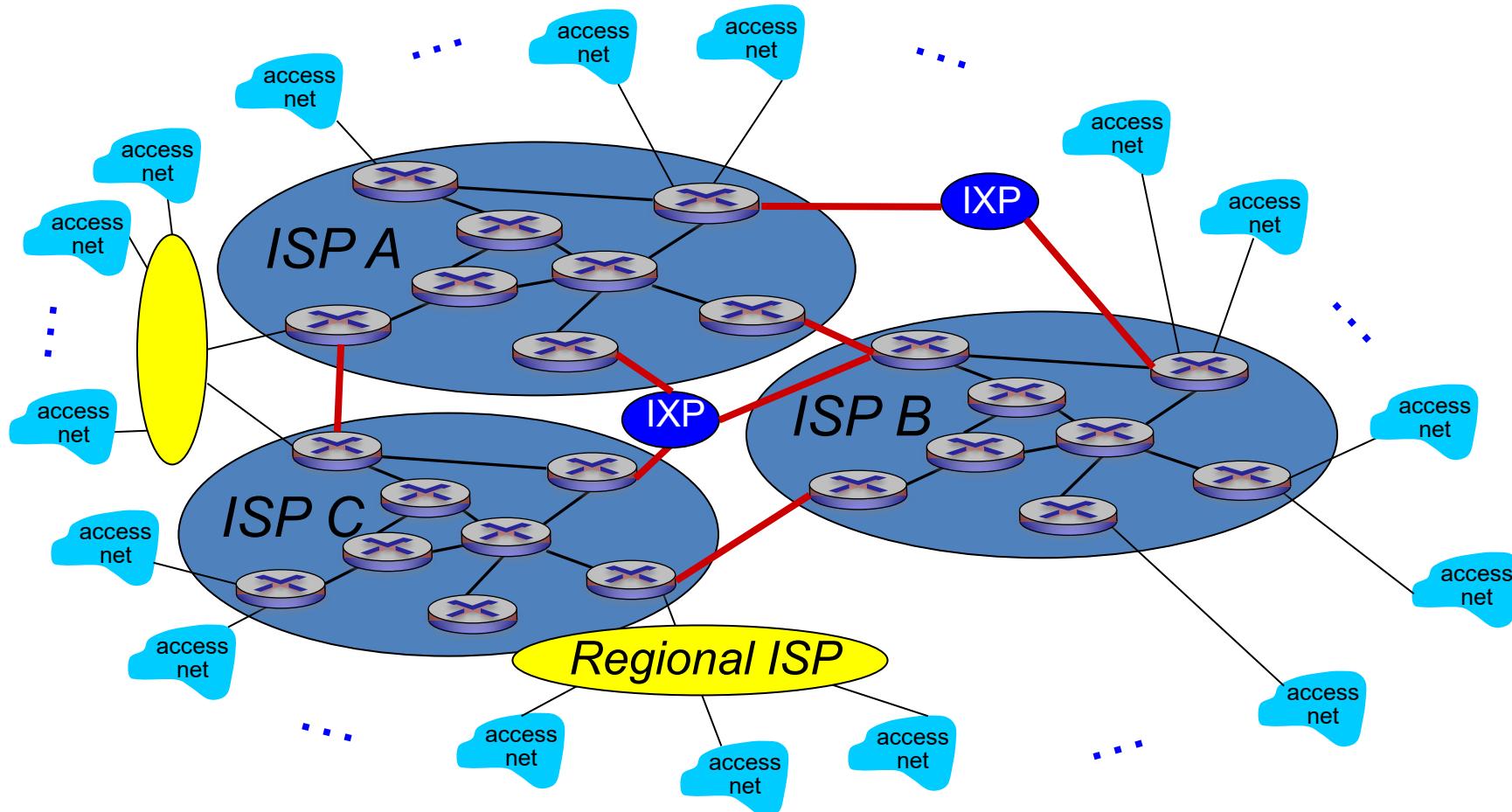
Internet structure: network of networks

But if one global ISP is viable business, there will be competitors
.... which must be interconnected



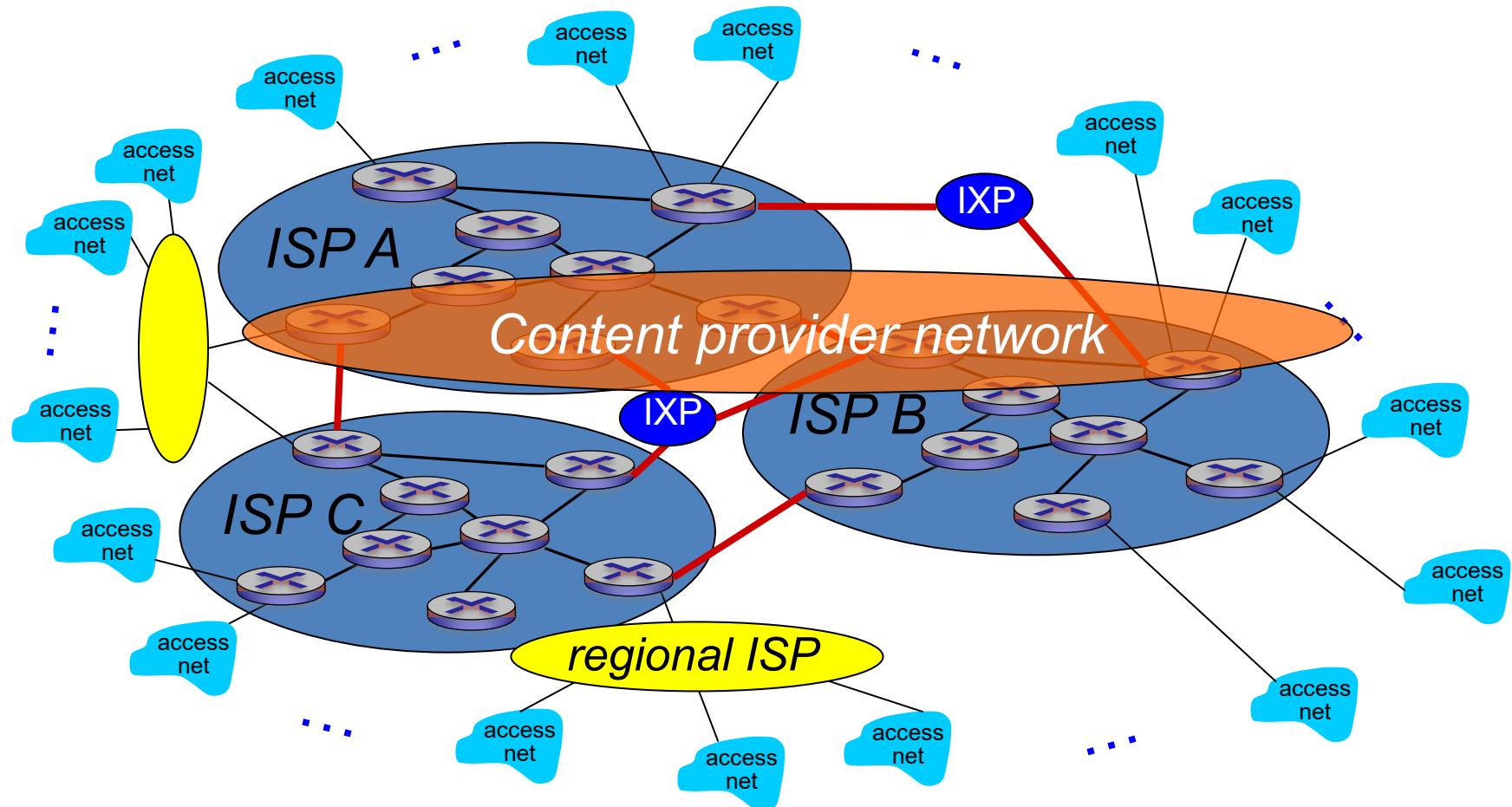
Internet structure: network of networks

... and regional networks may arise to connect access nets to ISPs

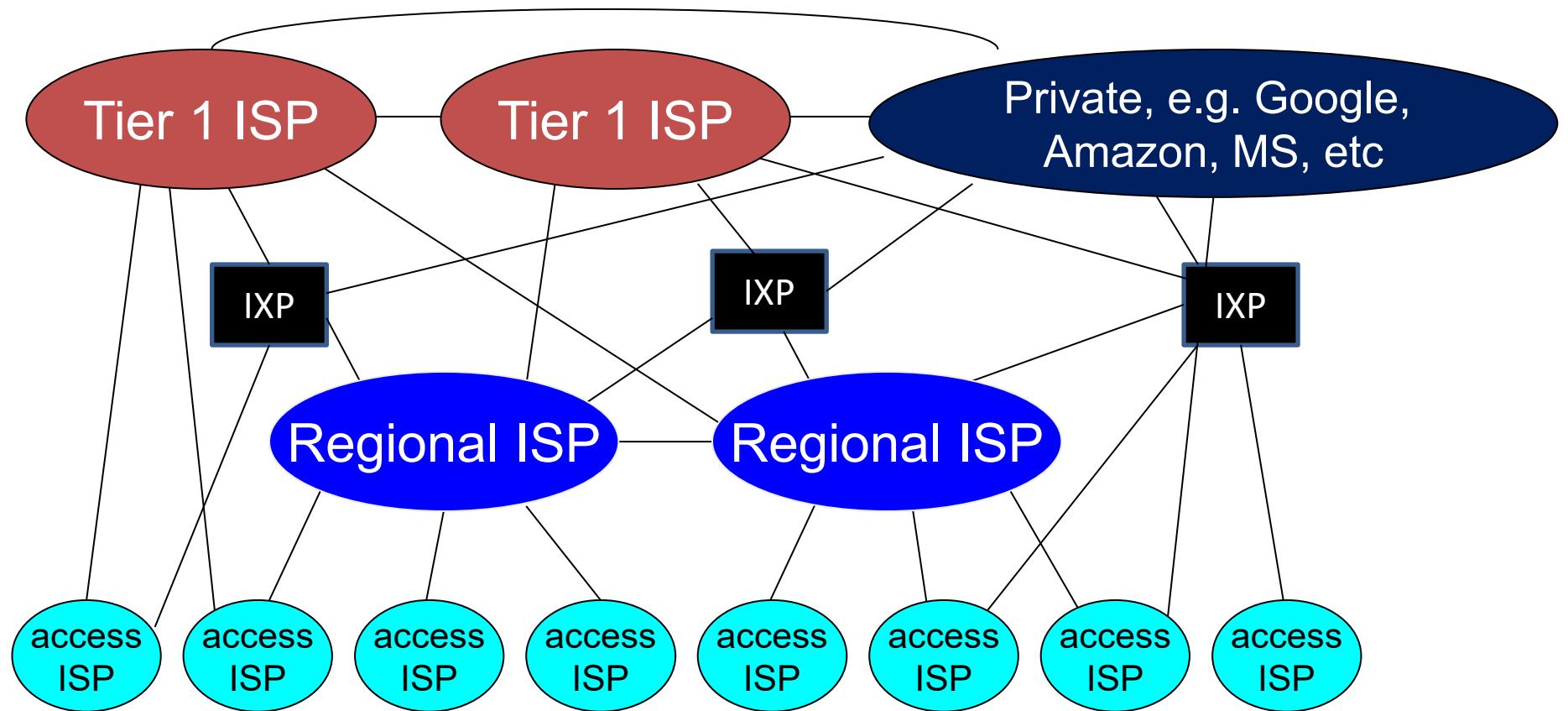


Internet structure: network of networks

... and content provider networks (e.g., Amazon, Google, Microsoft, Akamai) run their own networks, to bring services, content close to end users



Internet structure: network of networks

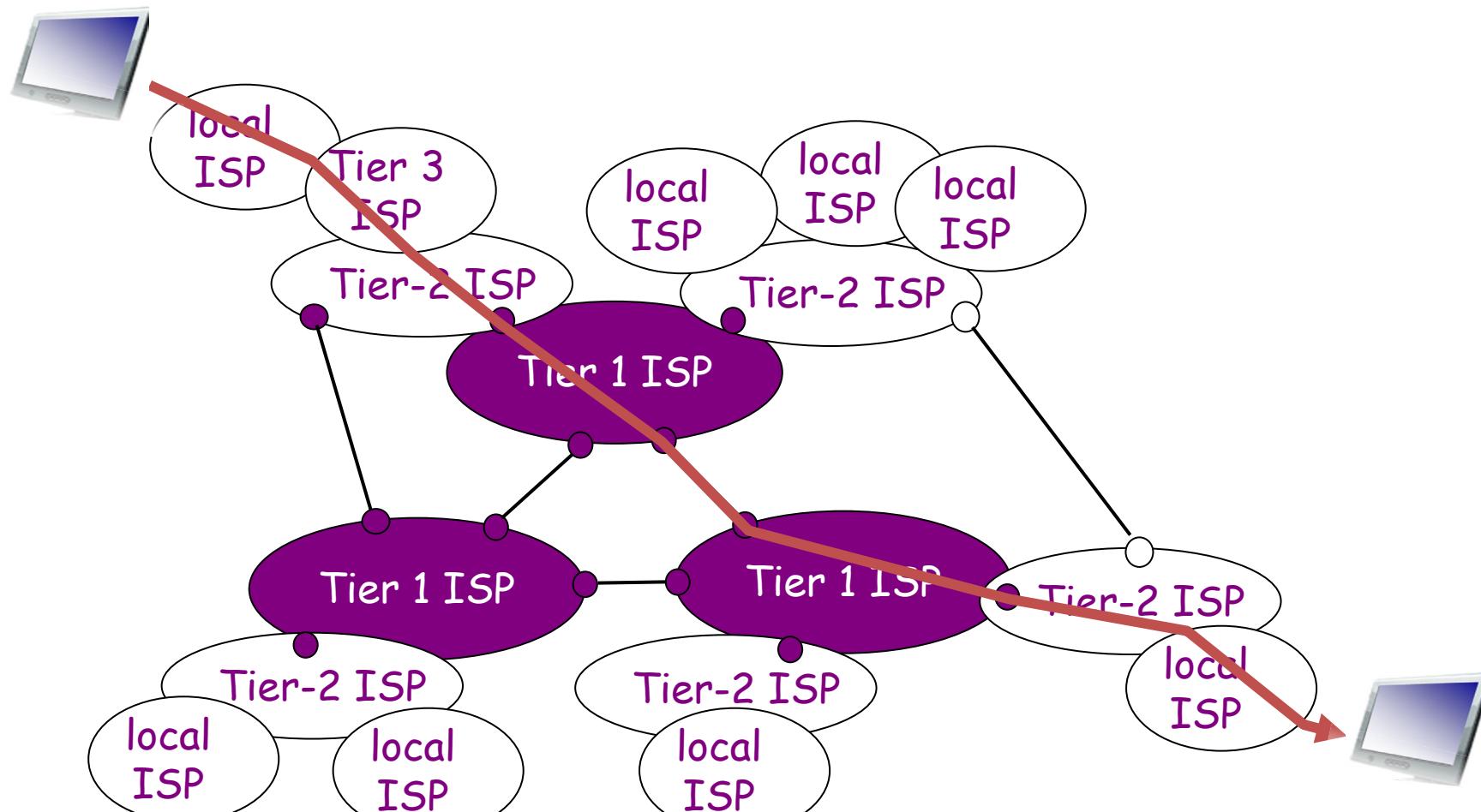


at center: small # of well-connected large networks

- “tier-1” commercial ISPs (e.g. AT&T, NTT, Telia), national & international coverage
- content provider network (e.g. Google): private network that connects its data centers to Internet, often bypassing tier-1, regional ISPs

Internet structure: network of networks

A packet passes through many networks!



... food for thought: implications/requirements?

Roadmap



1. Zooming into **the NW core**
 - Data transfer through packet switching
 - Performance:
 - delays (aka latency) & loss
 - throughput
2. Inter-connection structure:
 - access net, physical media
 - backbones, NAPs, ISPs
3. Security prelude

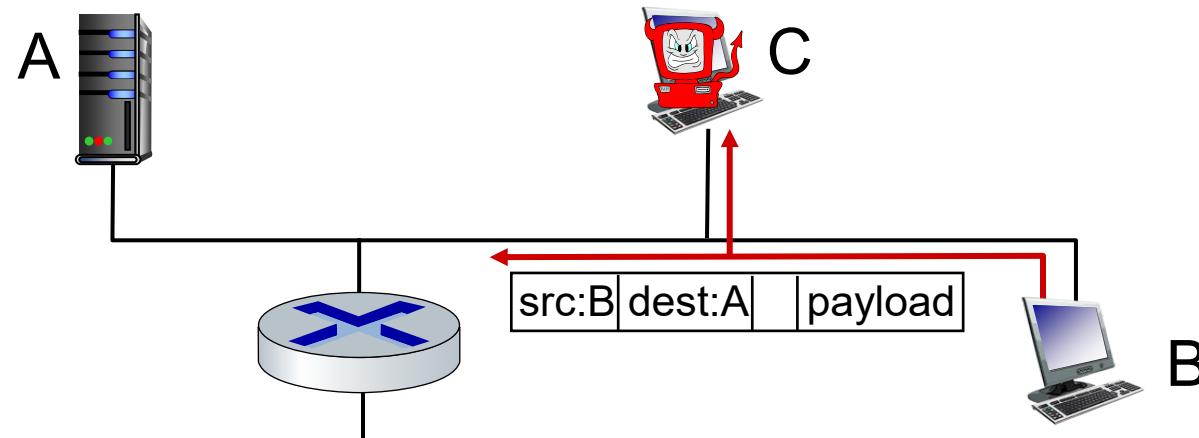
Network Security Prelude

- The field of network security is about:
 - how adversaries can attack computer networks (detection)
 - how to defend networks against attacks (mitigation)
 - how to design architectures that are immune to attacks (prevention)
- Internet not originally designed with (much) security in mind
 - *original vision*: “a group of mutually trusting users attached to a transparent network” ☺
 - Internet protocol designers playing “catch-up”
 - Security considerations in all layers!

Bad guys: packet interception

packet “sniffing”:

- broadcast media (shared Ethernet, wireless)
- promiscuous network interface reads/records all packets (e.g., including unencrypted passwords) passing by

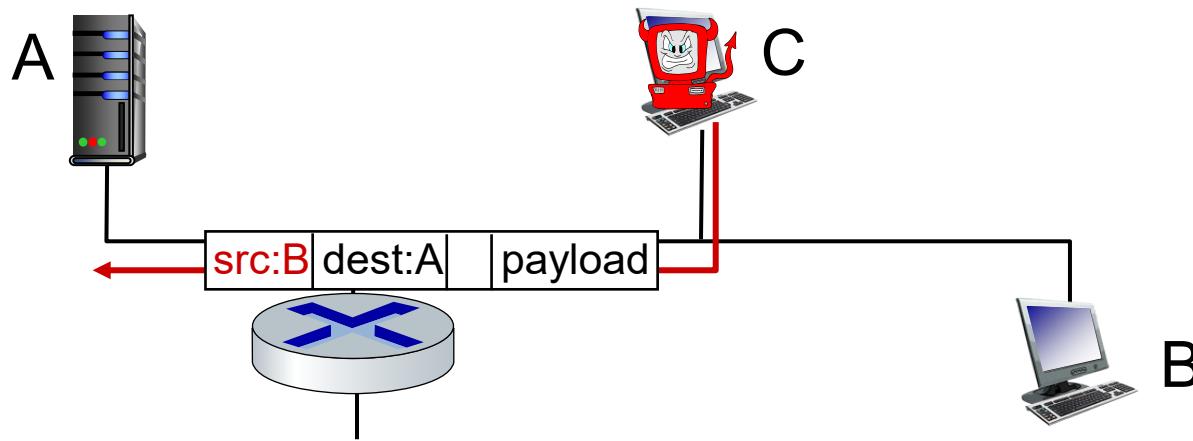


Wireshark software used for our labs is a (free) packet-sniffer

NOTE: be aware that it is inappropriate to use it outside the scope of the lab

Bad guys: fake identity

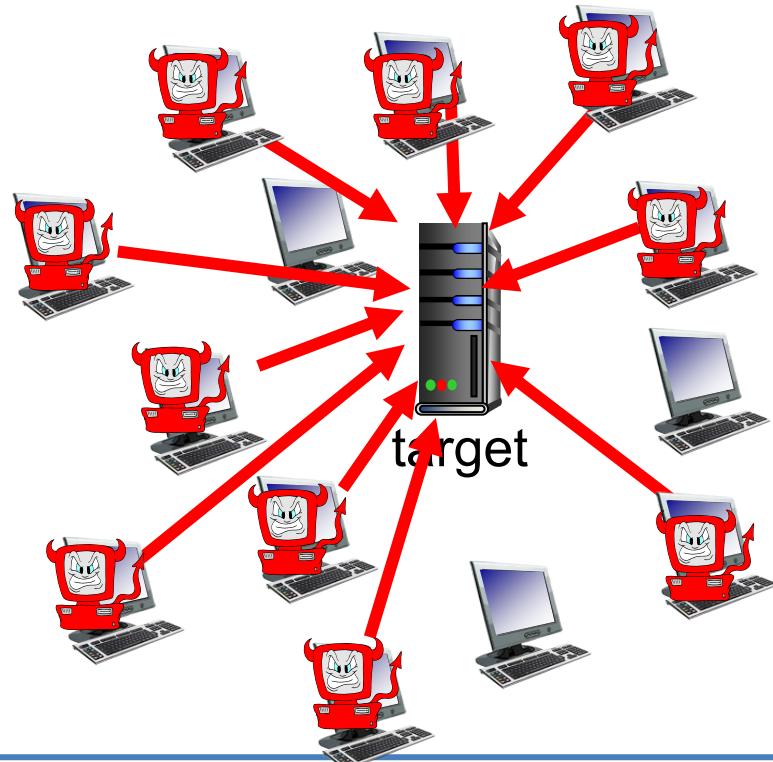
IP spoofing: send packet with false source address



Bad guys: denial of service

Denial of Service (DoS): attackers make resources (server, bandwidth) unavailable to legitimate traffic by overwhelming resource with bogus traffic

1. select target
2. break into hosts
around the network
(see botnet)
3. send packets to target
from compromised
hosts



Lines of defense:

- **authentication**: proving you are who you say you are
- **confidentiality**: via encryption
- **integrity checks**: digital signatures prevent/detect tampering
- **access restrictions**: e.g. password-protection
- **firewalls**: specialized “middleboxes” in access and core networks (filter packets, detect/react to attacks)

Chapter 1: Summary

Covered a lot of material!

- what's the Internet
- what's a protocol?
- protocol layers, service models
- Concepts of network edge, network
- Performance: delays (latency), throughput, loss
- NW structure: access nets (physical media), backbones, NAPs, ISPs
- Security concerns
- (history: read more in corresponding section, interesting & fun ☺)

In order to have:

- context, overview, “feeling ” of networking
- A point of reference for context in the more detailed discussions to come



Reading instructions

Kurose Ross book

Careful

1.2--1.5

Quick

the rest

CACM July 2019 - On The Hourglass Model - YouTube



Jun 26, 2019 - Uploaded by Association for Computing Machinery (ACM)

The **hourglass** model of layered systems architecture is a visual and conceptual representation of an approach ...

https://www.youtube.com/watch?v=L9s096_r_U

Extra Reading (optional)
Computer and Network Organization:
An Introduction,
by Maarten van Steen and Henk Sips,
Prentice Hall
(very good introductory book for
general engineering background)

Review questions for training

Review questions from Kurose-Ross book, chapter 1 (for basic study)

- R11, R12, R13, R16, 17, R18, R19, R20, R21, R22, R23, R24, R25, R28.

For further study (recommended but optional):

- delay analysis in packet switched networks:

<https://www.comm.utoronto.ca/~jorg/teaching/ece466/07/material/466-SimpleAnalysis.pdf>

- Morten Bay (2019) Hot potatoes and postmen: how packet switching became ARPANET's greatest legacy, Internet Histories, 3:1, 15-30,

<https://doi.org/10.1080/24701475.2018.1544726>

More example/review questions for training

- Suppose there is exactly one packet switch between a sending host and a receiving host. The transmission rates between the sending host and the switch and between the switch and the receiving host are R_1 and R_2 , respectively. Assuming that the switch uses store-and-forward packet switching, what is the total end-to-end delay to send a packet of length L ? (Ignore queuing, propagation delay, and processing delay.)
- What is the role of output queue in packet switching?

How long does it take a packet of length 1,000 bytes to propagate over a link of distance 2,500 km, propagation speed $2.5 \cdot 10^8$ m/s, and transmission rate 2 Mbps? More generally, how long does it take a packet of length L to propagate over a link of distance d , propagation speed s , and transmission rate R bps? Does this delay depend on packet length? Does this delay depend on transmission rate?

- Does layering have any disadvantages?
- What is an application-layer message? A transport-layer segment? A network-layer datagram? A link-layer frame?
- Which layers in the Internet protocol stack does a router process? Which layers does a link-layer switch process? Which layers does a host process?



Course on Computer Communication and Networks

Lecture 3

Chapter 2: Application-layer, basic application protocols

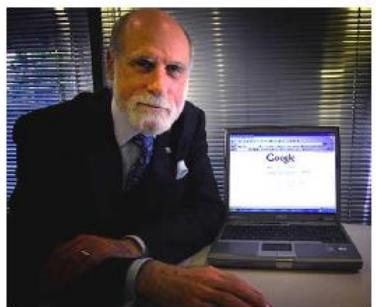
EDA344/DIT 423/LEU062

Lecturer: Hans-Martin Heyn

Based on the book Computer Networking: A Top Down Approach, Jim Kurose, Keith Ross, Addison-Wesley.

Recall: on the “critical path” of Internet’s evolution

1974



TCP / IP defined by Vint Cerf & Bob Kahn



2004: both received the Turing Award

1971



Ray Tomlinson creates first email program

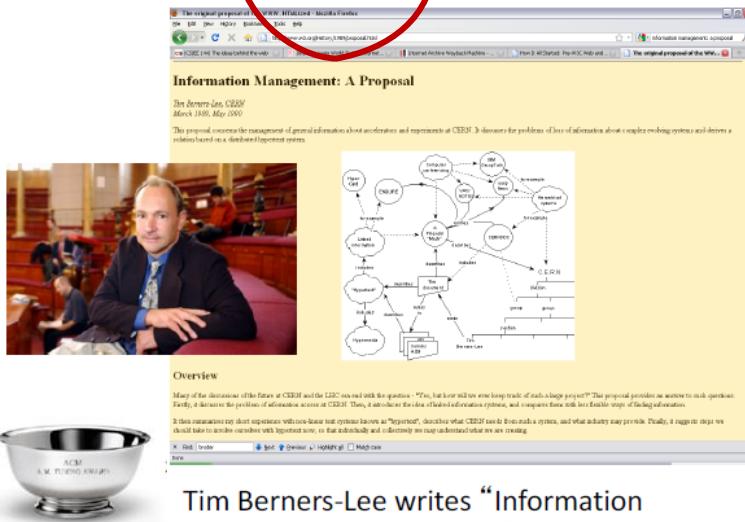
1984



Paul Mockapetris introduces DNS

...on the “critical path” of the Internet’s evolution (cont.)

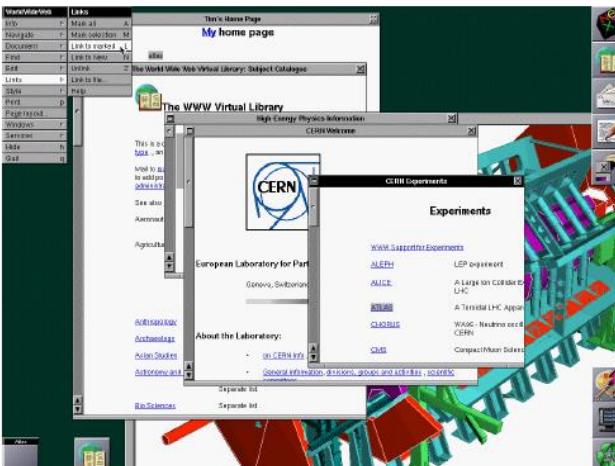
1989 – The Web Emerges



Tim Berners-Lee writes “Information Management: A proposal” at CERN
2016 Turing award

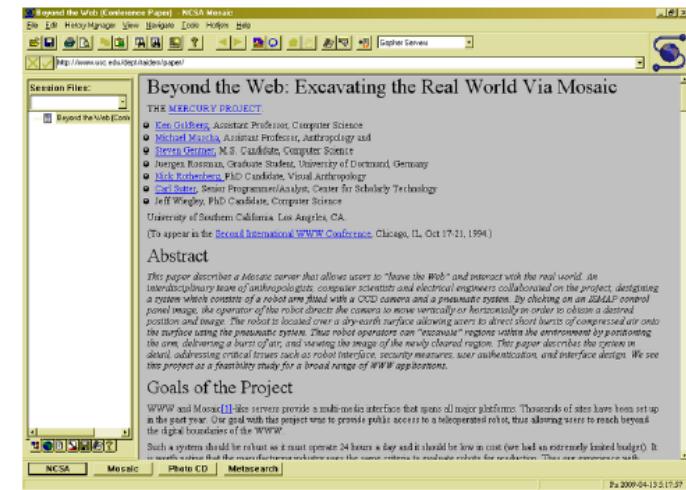
Watch this 3 min video:
A brief history of the World Wide Web
https://www.youtube.com/watch?v=sSqZ_hJu9zA

1990



First browser developed at CERN

1993



Mosaic became the first graphical browser

1995+

Amazon arrives and the commercialization of the web begins



Amazon circa 1999

This lecture’s goals -- learn about:

- key aspects of app-layer protocols
- these basic app-layer protocols (HTTP, SMTP, DNS)

Internet protocol stack layers&protocols

Application: protocols supporting *network applications*

http (*web*), smtp (*email*), p2p, streaming, CDN, ...

Transport: process2process (end2end) data transfer

UDP (user Datagram Prot.), **TCP** (Transmission Control Prot.)

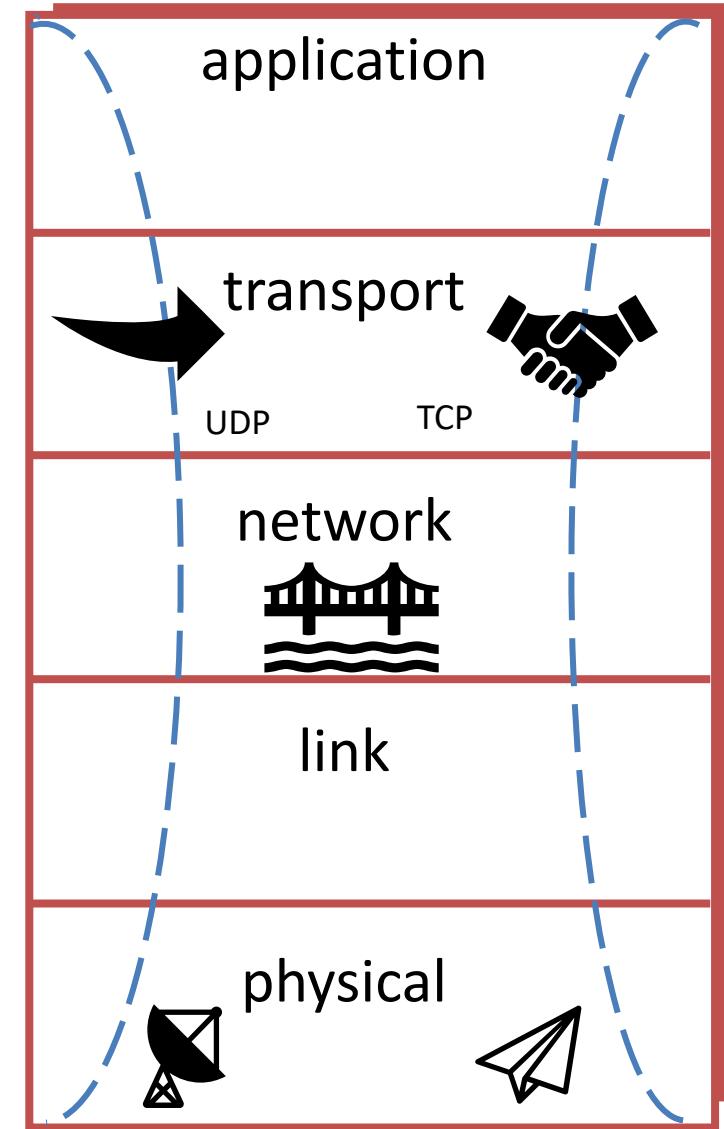
Network: routing of data-packets, connecting nodes at different physical networks

Universal addressing to bridge NWs with other addressing formats (IP**), routing protocols**

Link: data transfer between directly connected nodes

Ethernet, WiFi, ...

Physical: bit-transmission on the physical medium between directly connected nodes



Layering: The OSI Reference Model

ISO (International Standards Organization) defined the OSI (Open Systems Interconnect) model to help vendors create interoperable network implementation

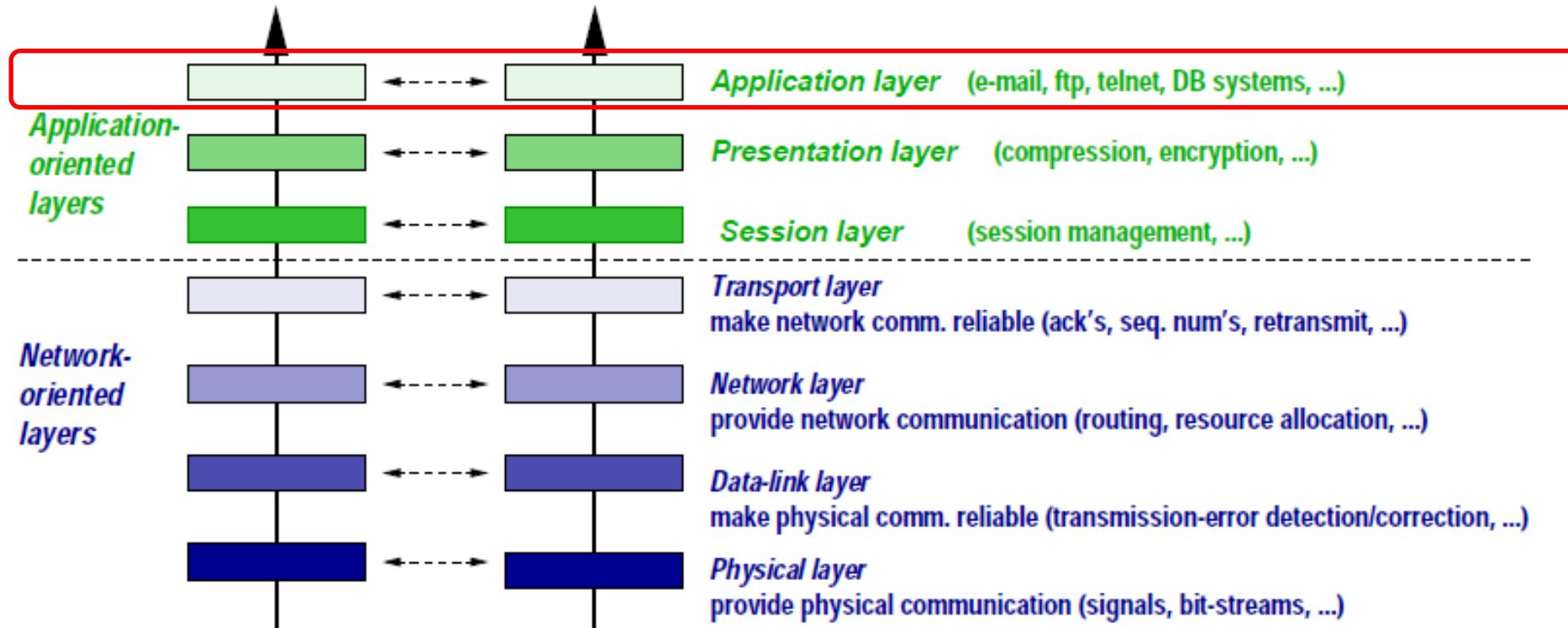


Fig. Steen, Sips : Computer and Network organization

``X dot'' series (X.25, X. 400, X.500) OSI model implementation (protocol stack)

Creating a network app

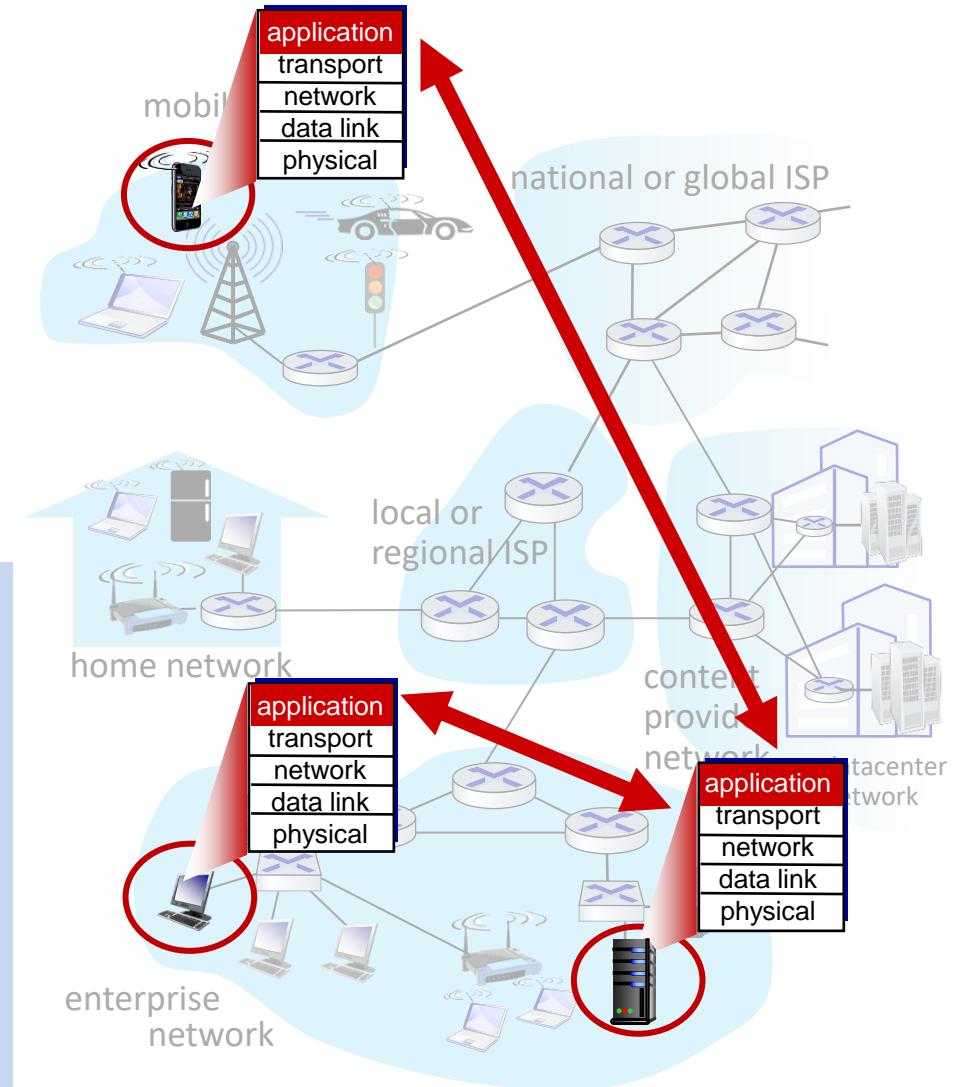
write programs (i.e. communicating processes) that:

- run on (different) end systems
- communicate over network
- e.g., web server software communicates with browser software

... without need to (re)build the network-core functions

Application-layer protocols...

1. ...are only one “piece” of an application -others are e.g. user agents.
 - Web: browser <- user agent
 - E-mail: mail reader <- user agent
 - streaming audio/video: Netflix / Spotify <- user agent
2. ...**define** messages exchanged and actions taken & **use services** provided by lower layer protocols



An application-layer protocol defines:

- types of messages exchanged,
 - e.g., request, response
- message syntax:
 - what fields in messages & how fields are delineated
- message semantics
 - meaning of information in fields
- rules for when and how processes send & respond to messages

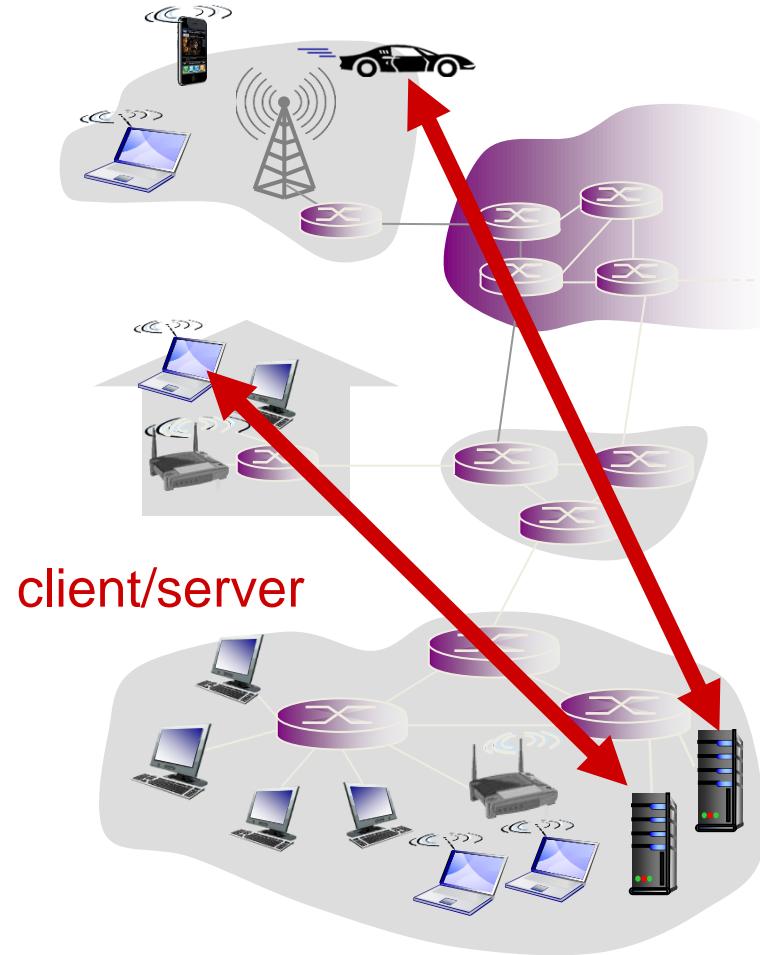
open protocols:

- defined in RFCs, everyone has access to protocol definition
- allows for interoperability
- e.g., HTTP, SMTP

proprietary protocols:

- e.g., Zoom, Netflix, WhatsApp, Microsoft Exchange

Client-server applications architecture



server:

- always-on
- permanent host address
- clusters of servers for scaling

clients:

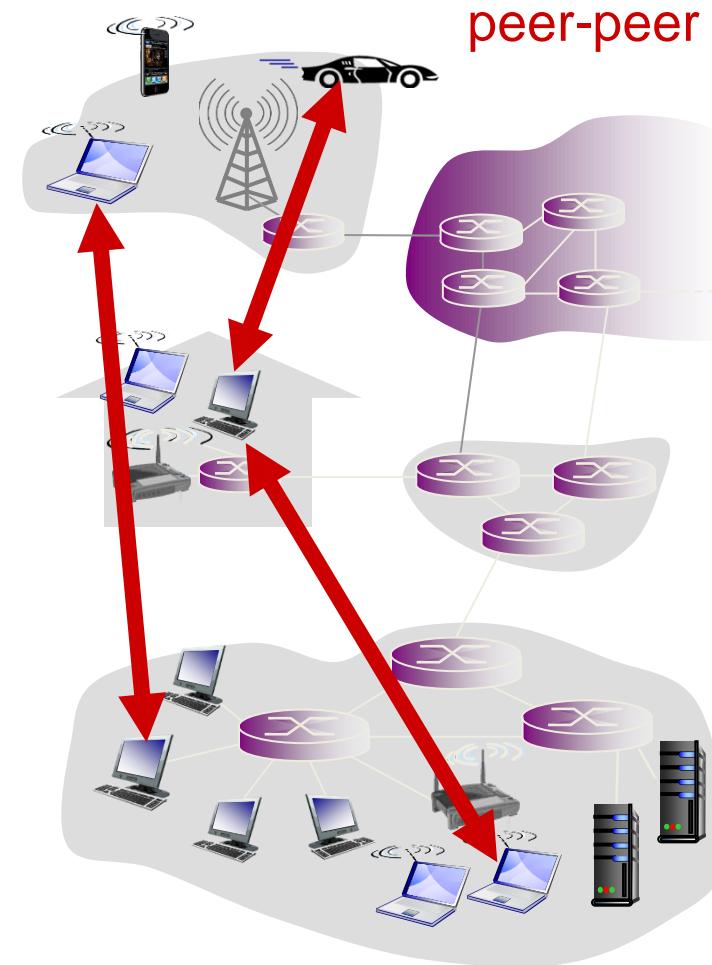
- communicate with server
- may be intermittently connected
- may have dynamic host addresses

e.g., http, ftp, Netflix content delivery, most social networks

Peer2Peer applications architecture

- no always-on server
- peers request service from other peers, provide service in return
 - *self scalability* – new peers bring service capacity, and new service demands
- peers are intermittently connected and may change addresses
 - complex management

e.g., p2p file sharing, bitcoin, Windows 10/11 Update Service



Roadmap



- Addressing
 - Applications needs from transport layer
 - Http
 - General description and functionality
 - About maintaining state
 - Performance:
 - Persistence, pipelining
 - Caching and proxies
 - SMTP (POP, IMAP)
 - DNS

i.e. key aspects of basic application protocols

(more application-layers protocols : later in the course)



Processes communicating

- process*: program running within a host
- within same host, two processes communicate using **inter-process communication** (defined by OS)
 - processes in different hosts communicate by exchanging **messages**

clients, servers

client process: process that initiates communication

server process: process that waits to be contacted

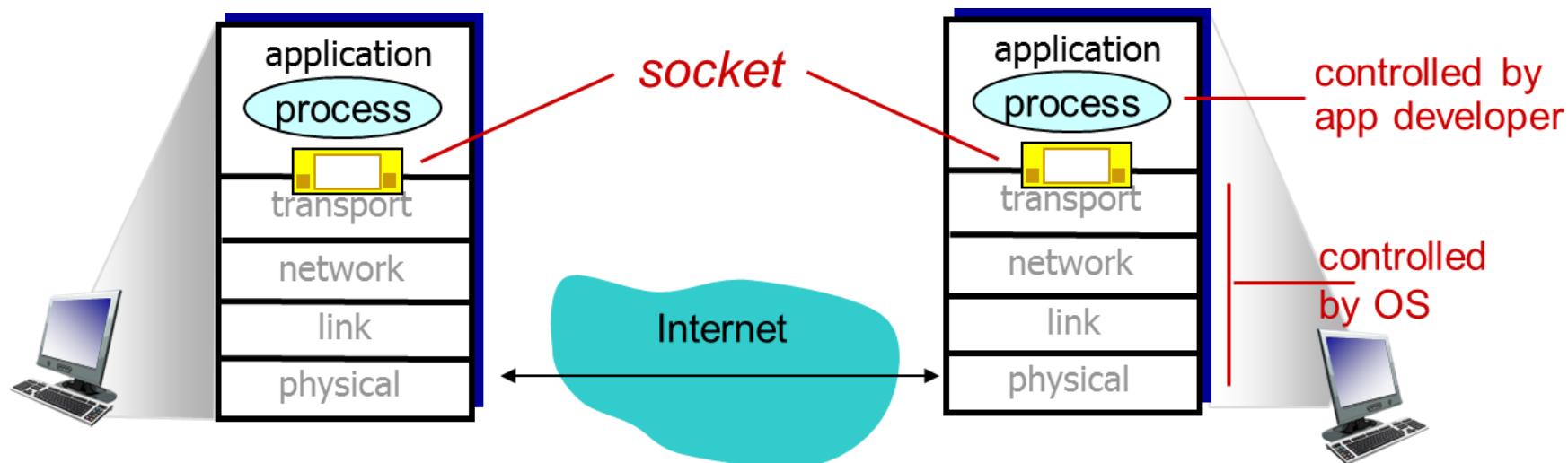
Addressing, sockets

socket: Internet's API (resembles a “door”)

- 2 processes communicate via 2 sockets by:
 - sending data into a socket
 - reading data out of a socket

Q: how does a process “address” the other process with which it wants to communicate? (i.e., its socket)

- **IP address** of host running other process
- “**port number**” - allows receiving host to determine to which local process the message should be delivered



Roadmap

- Addressing
- Applications needs from transport layer
- Http
 - General description and functionality
 - About maintaining state
 - Performance:
 - Persistence, pipelining
 - Caching and proxies
- SMTP (POP, IMAP)
- DNS



What transport service can an app care about?

data reliability

- some apps (e.g., file transfer, web transactions) require 100% reliable data transfer
- other apps (e.g., audio) can tolerate some loss

latency

- some apps (e.g., Internet telephony, interactive games) require low delay to be “effective”

throughput

- some apps (e.g., multimedia) require minimum amount of throughput to be “effective”
- other apps (“elastic apps”) make use of whatever throughput they get

security

- encryption, data integrity, ...

Transport service requirements: common apps

application	data loss	throughput	time sensitive?
file transfer/download	no loss	elastic	no
e-mail	no loss	elastic	no
Web documents	no loss	elastic	no
real-time audio/video	loss-tolerant	audio: 5Kbps-1Mbps video:10Kbps-5Mbps	yes, 10's msec
streaming audio/video	loss-tolerant	same as above	yes, few secs
interactive games	loss-tolerant	Kbps+	yes, 10's msec

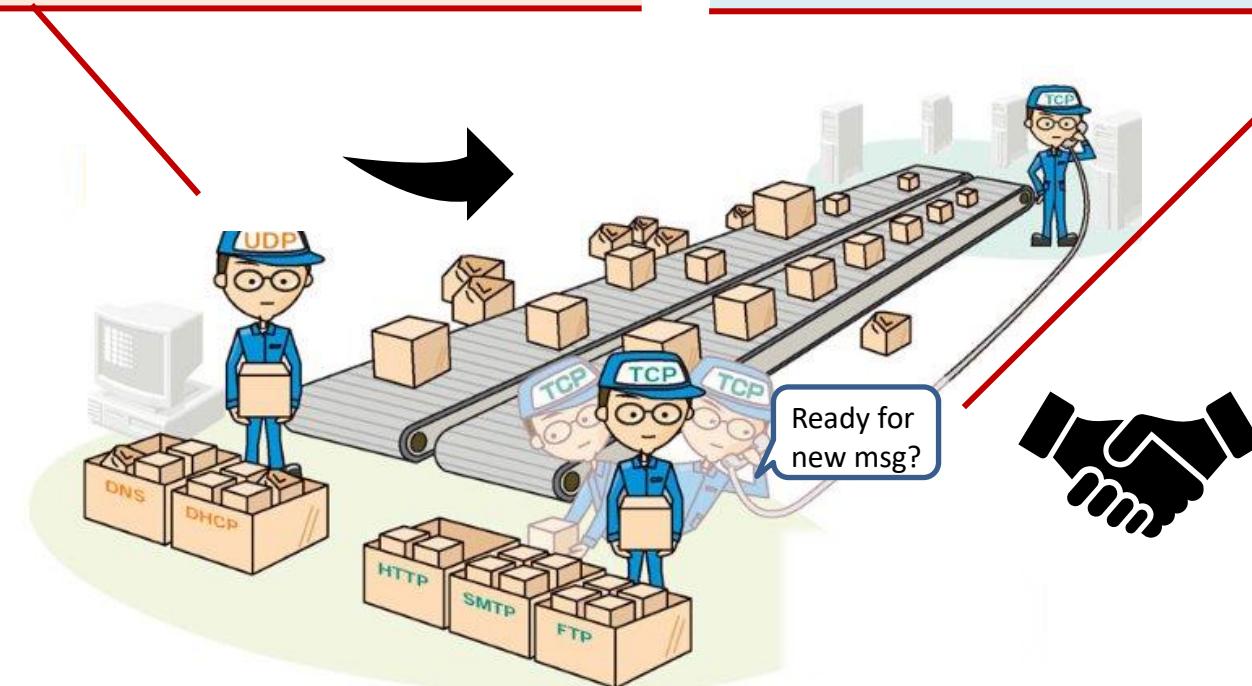
Services to upper layer by Internet transport protocols

UDP's service:

- **Connectionless:** *Unreliable transport* between sending and receiving process
 - “*best-effort*” delivery
- **does *not* provide:** reliability, throughput, security guarantees

TCP's service:

- **connection-oriented:** *reliable transport* between sending and receiving process
 - correct, in-order data delivery, synch required between client, server
- **does *not* provide:** throughput, security guarantees



Internet transport protocols services in apps

application	application layer protocol	transport protocol
file transfer/download	FTP [RFC 959]	TCP
e-mail	SMTP [RFC 5321]	TCP
Web documents	HTTP 1.1 [RFC 7320]	TCP
Internet telephony	SIP [RFC 3261], RTP [RFC 3550], or proprietary	TCP or UDP
streaming audio/video	HTTP [RFC 7320], DASH	TCP
interactive games	WOW, FPS (proprietary)	UDP or TCP

Small Excursion: Securing TCP

Vanilla TCP & UDP sockets:

- no encryption
- cleartext passwords sent into socket traverse Internet in cleartext (!)

Transport Layer Security (TLS)

- provides encrypted TCP connections
- data integrity
- end-point authentication

TLS implemented in application layer

- apps use TLS libraries, that use TCP in turn

TLS socket API

- cleartext sent into socket traverse Internet *encrypted*

Roadmap

- Addressing
- Applications needs from transport layer
- **Http**
 - General description and functionality
 - About maintaining state
 - Performance:
 - Persistence, pipelining
 - Caching and proxies
- SMTP (POP, IMAP)
- DNS



1989 – The Web Emerges

Information Management: A Proposal

Tim Berners-Lee writes “Information Management: A proposal” at CERN
2016 Turing award

Watch this 3 min video:
A brief history of the World Wide Web
https://www.youtube.com/watch?v=sSqZ_hJu9zA

Web and HTTP

First, some jargon...

- *web page* consists of *objects*
- object can be HTML file, JPEG image, Java applet, audio file,...
- web page consists of *base HTML-file* which includes *several referenced objects*
- each addressable by a *URL*, e.g.,

`https://www.chalmers.se/_next/static/media/chalmers.26fdad12.svg`

host name path name

CHALMERS

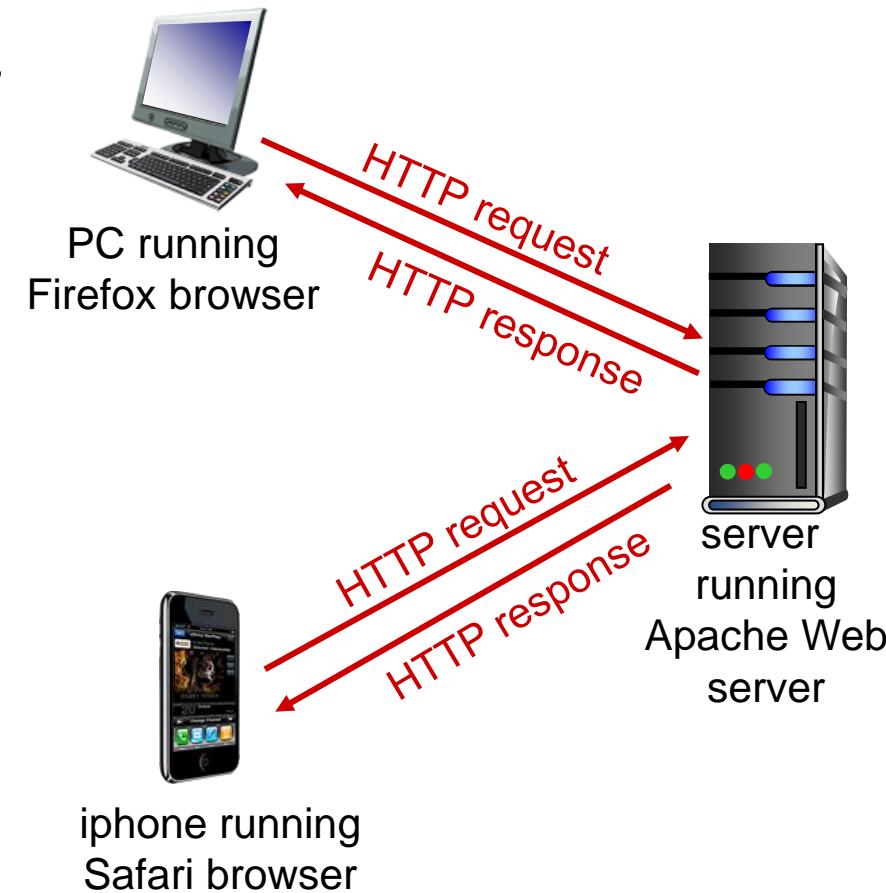
HTTP: hypertext transfer protocol overview

Web's application layer protocol

- *http client*: web browser; requests, receives, displays Web objects
- *http server*: Web server sends objects

uses TCP:

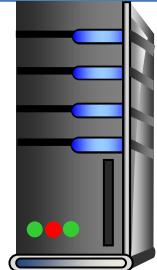
1. client initiates TCP connection to server,
port 80
2. server accepts TCP connection
3. HTTP messages exchanged
4. TCP connection closed



Elementary http example: user enters URL



e.g., <https://www.chalmers.se/en/departments/cse/Pages/default.aspx>
(contains some text, references to 10 jpeg images)



1. **http client** initiates TCP connection to http server (process) at www.chalmers.se. Port 80 is default for http server.

2. **http server** at host www.chalmers.se waiting for TCP connection at port 80, “accepts” connection, notifying client

3a. client sends http *request message* (containing URL) into TCP connection socket

3b. server receives request, forms *response message* with requested object [/en/departments/cse/Pages/default.aspx](https://www.chalmers.se/en/departments/cse/Pages/default.aspx), sends msg into socket

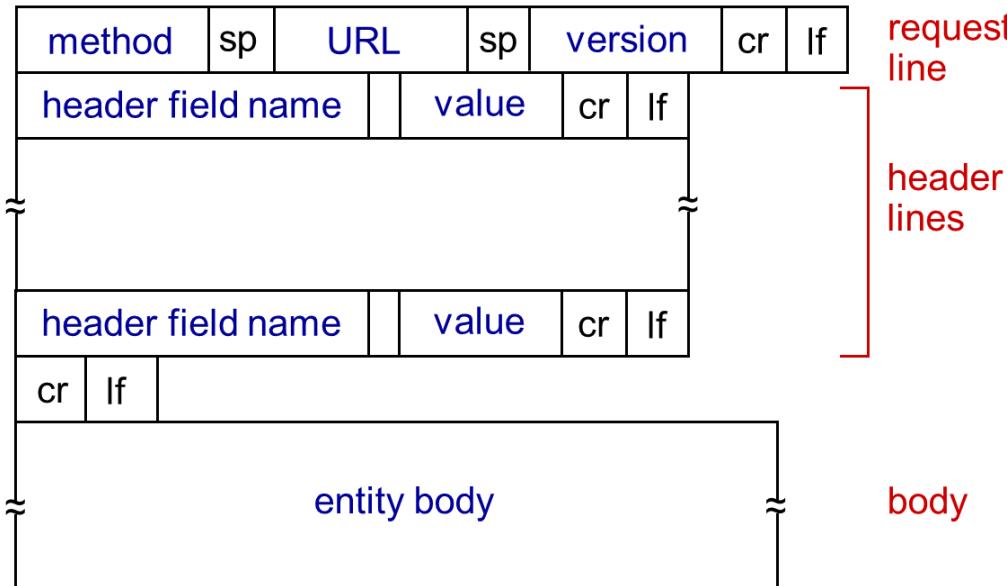
3c. client receives response msg with file, displays html. Parsing html file, finds 10 referenced jpeg objects

4. server closes TCP connection.

time

Steps 1-5 repeated for each of 10 jpeg objects

HTTP request message: general format



Methods:

- GET
- POST: e.g., data to some input form
- PUT: uploads file in body to path specified in URL field
- HEAD: asks server to send only header

HTTP response message

status line
(protocol
status code
status phrase)

header
lines

```
HTTP/1.1 200 OK\r\nDate: Sun, 23 Sep 2018 20:09:20 GMT\r\nServer: Apache/2.0.52 (CentOS) \r\nLast-Modified: Mon, 30 Oct 2017 17:00:02 GMT\r\nETag: "17dc6-a5c-bf716880"\r\nAccept-Ranges: bytes\r\nContent-Length: 2652\r\nKeep-Alive: timeout=10, max=100\r\nConnection: Keep-Alive\r\nContent-Type: text/html; charset=ISO-8859-1\r\n\r\n
```

validation,
for eg caching

data data data data ...

data, e.g.,
requested
HTML file

200 OK: request succeeded, requested object in this msg

301 Moved Permanently: requested object moved, new location
specified later in this message (Location:)

400 Bad Request: request message not understood

404 Not Found: requested document not found on this server

505 HTTP Version Not Supported

* Check out the online interactive
exercises for more examples:
http://gaia.cs.umass.edu/kurose_ross/interactive/

Roadmap

- Addressing
- Applications needs from transport layer
- Http
 - General description and functionality
 - About maintaining state**
 - Performance:
 - Persistence, pipelining
 - Caching and proxies
- SMTP (POP, IMAP)
- DNS



HTTP is “stateless”

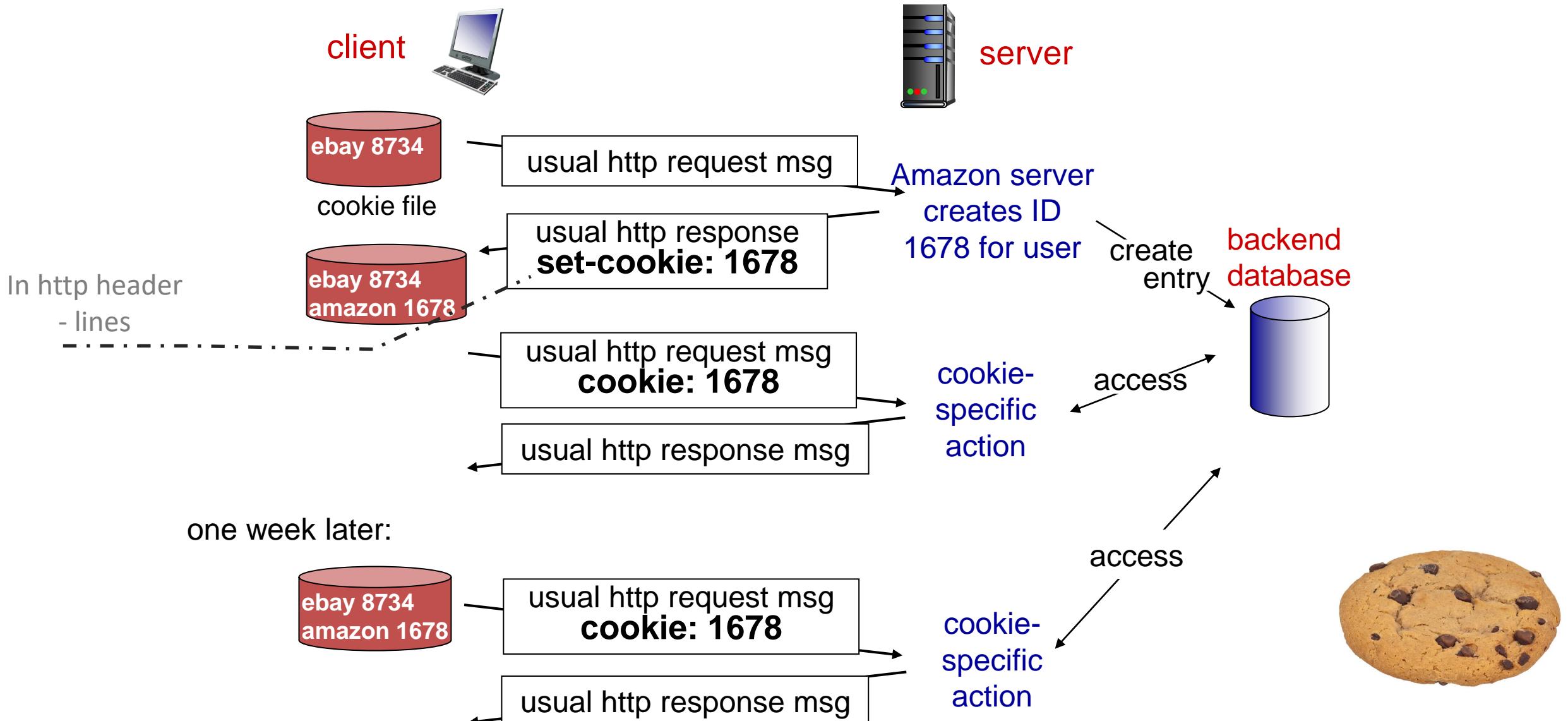
server maintains no information about client requests etc

- protocols that maintain “state” are complex!
- past history (state) must be maintained
- if server/client crashes, their views of “state” may be inconsistent, must be reconciled

Q: how do web applications keep state though?



Cookies: keeping “state”



Cookies (continued)

cookies can bring:

- authorization
- shopping carts
- recommendations
- user session state (eg web-email)

aside

Cookies and privacy:

- cookies permit sites to learn a lot about us
- we may supply name and e-mail to sites
- search engines use cookies to learn yet more
- third party persistent cookies (tracking cookies) allow common identity (cookie value) to be tracked across multiple web sites

Some Q&A about cookies

(English) <https://www.pts.se/en/english-b/regulations2/legislation/electronic-communications-act/q/>

(Swedish) <https://www.pts.se/sv/privat/internet/integritet/kakor-cookies/>

Roadmap

- Addressing
- Applications needs from transport layer
- Http
 - General description and functionality
 - About maintaining state
 - Performance:**
 - Persistence, pipelining
 - Caching and proxies
- SMTP (POP, IMAP)
- DNS



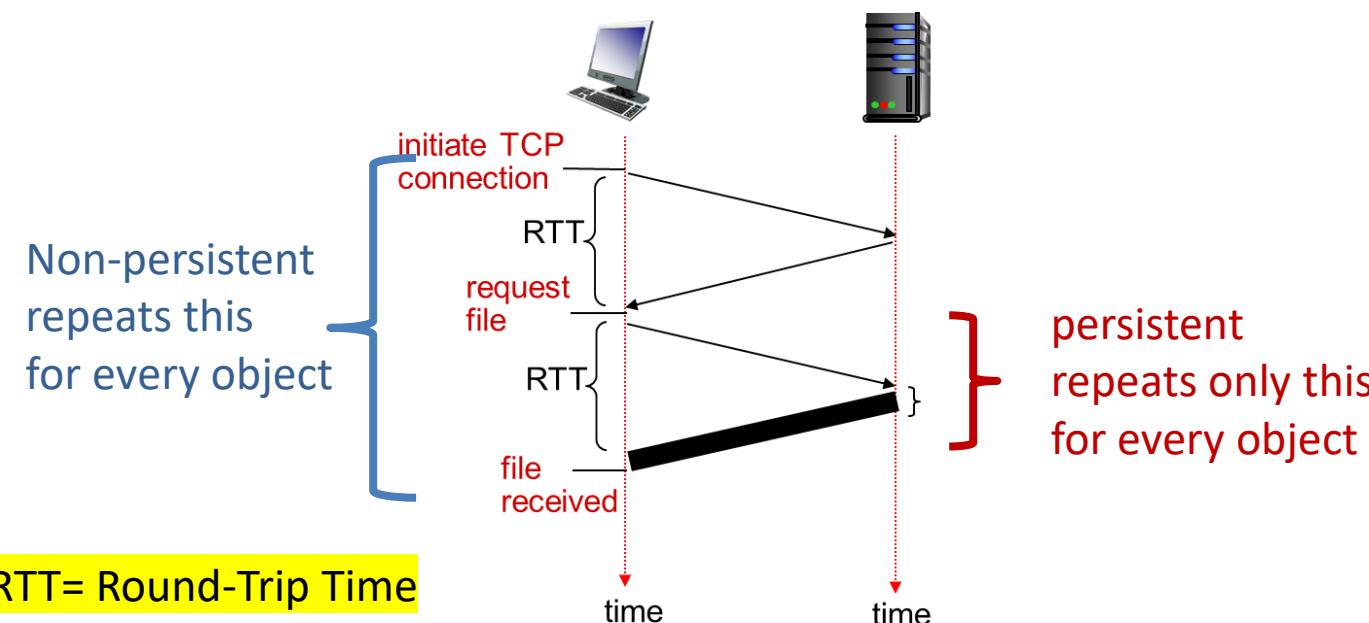
Non-persistent and persistent http

Non-persistent (http 1.0)

- server parses request, responds, closes TCP connection
- *non-persistent HTTP response time = 2RTT + file transmission time*
- new TCP connection for each object => extra RTT overhead per object

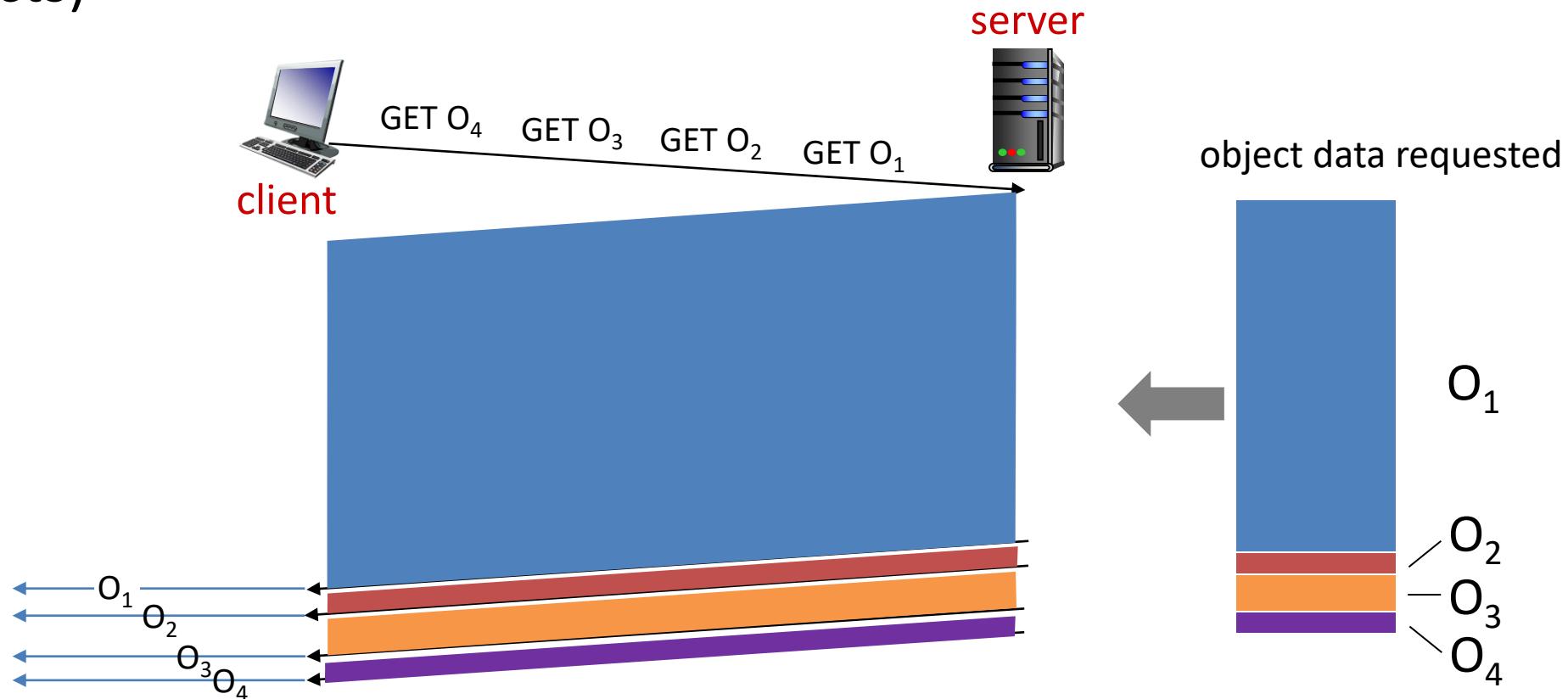
Persistent (http 1.1)

- **on same TCP connection:** server parses request, responds, parses new request,...
- Client sends requests for all referenced objects in the base HTML;
 - Less overhead per object
- Objects are fetched sequentially



Problem: HOL (head-of-line) blocking

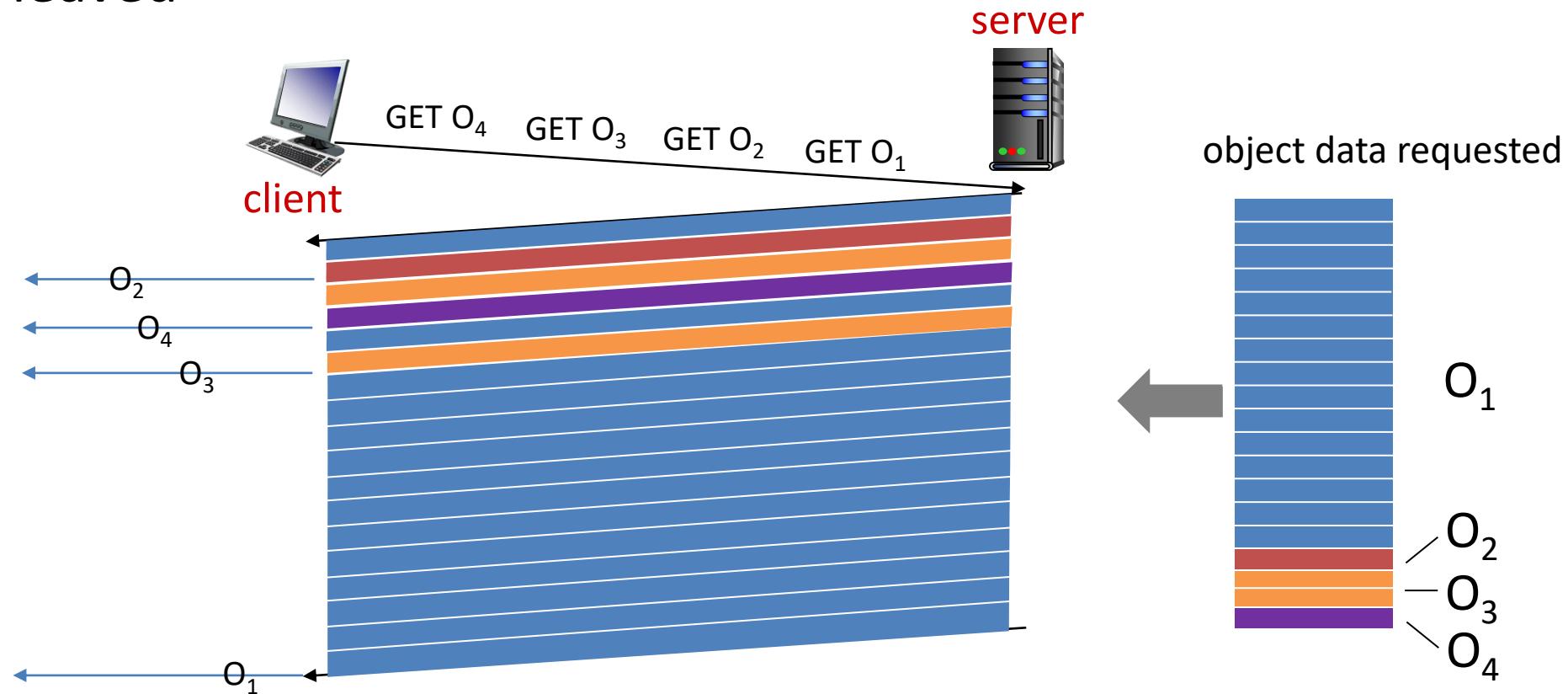
HTTP 1.1: client requests 1 large object (e.g., video file, and 3 smaller objects)



objects delivered in order requested: O_2 , O_3 , O_4 wait behind O_1

Improvement: HTTP/2 : mitigating HOL blocking

HTTP/2 (RFC 7540): objects divided into frames, frame transmission interleaved



O₂, O₃, O₄ delivered quickly, O₁ slightly delayed

About HTTP/2 (and /3)

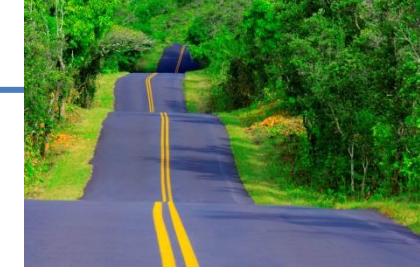
- major revision of HTTP, derived also from Google's SPDY
 - on top of Google's QUIC transport/app layer protocol (instead of TCP)
 - +TLS (https: http over TLS, port 443) encoding, priorities
- supported by many browsers
- proposed successor: HTTP/3



Better explanation needs cross-layer issues' understanding => later in the course

Roadmap

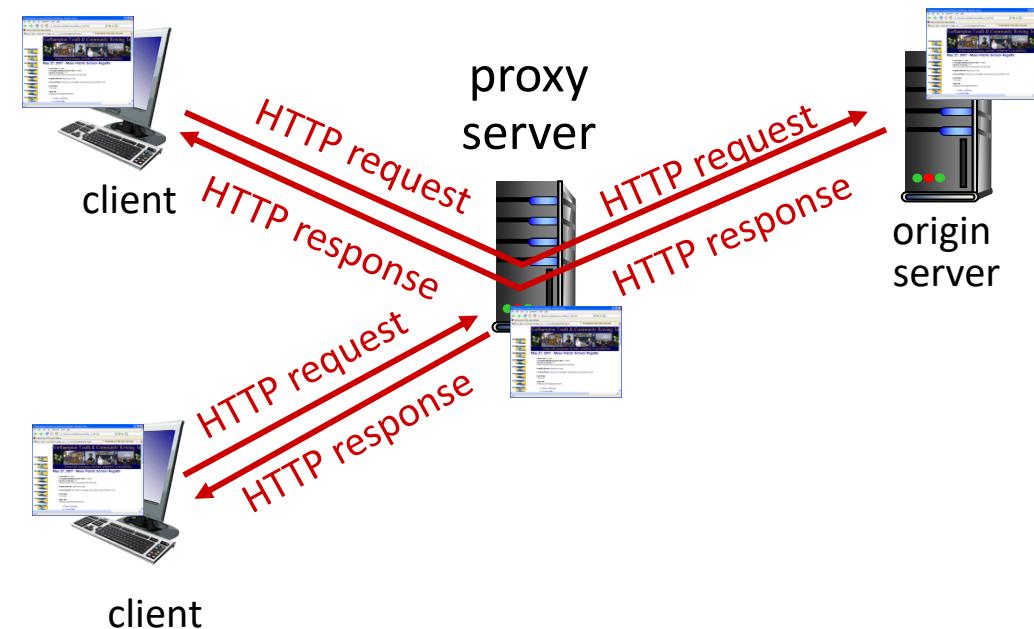
- Addressing
- Applications needs from transport layer
- Http
 - General description and functionality
 - About maintaining state
 - Performance:
 - Persistence, pipelining
 - Caching and proxies
- SMTP (POP, IMAP)
- DNS



Web Caches (proxy server)

Goal: satisfy client request without involving origin server

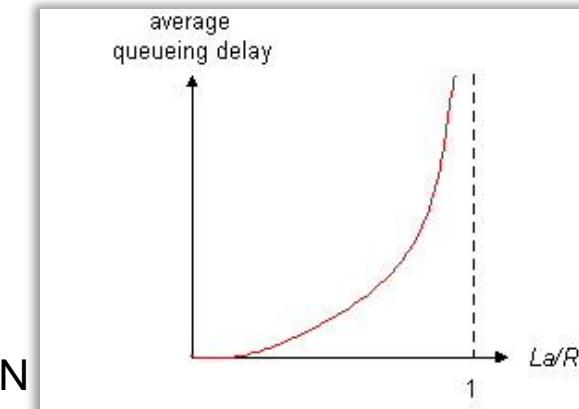
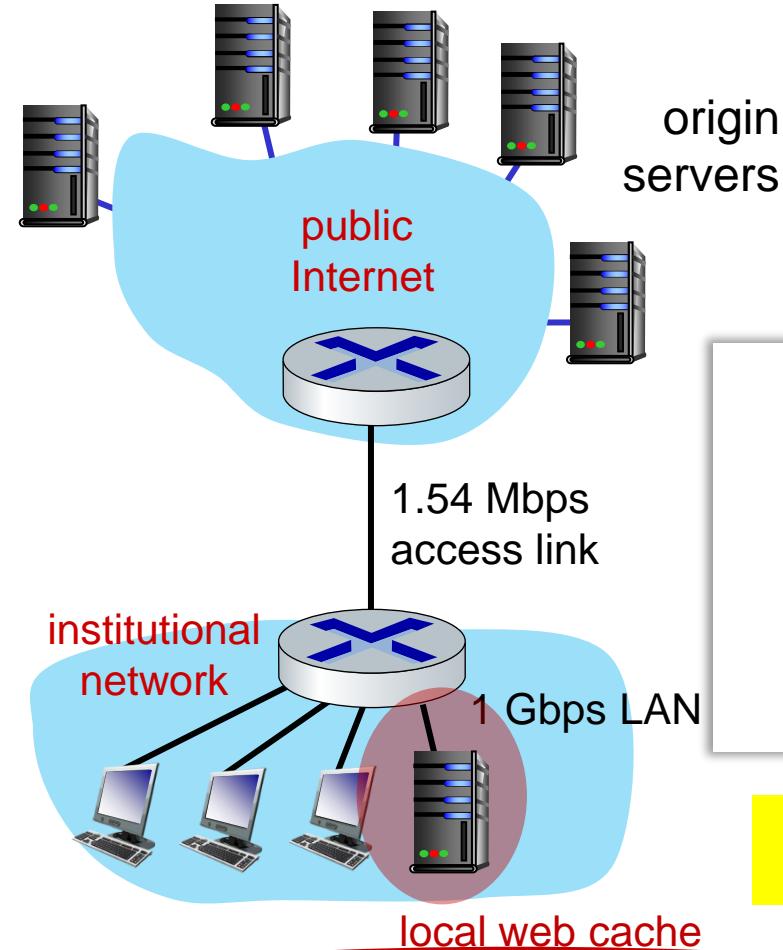
- user configures browser: Web accesses via web cache
- client sends all http requests to web cache; the cache(proxy) server acts as usual caches do
- Hierarchical, cooperative caching, ICP: Internet Caching Protocol (RFC2187)



Why Web Caching?

Cache is close to client (installed by ISP) =>

- smaller response time
- decrease traffic to distant servers
 - link to/form institutional/local ISP network can be bottleneck
- Important for big data applications (e.g. video,...)
 - Internet is dense with caches
 - enables **lower-cost** effective deliver content (compared to upgrades of access link)

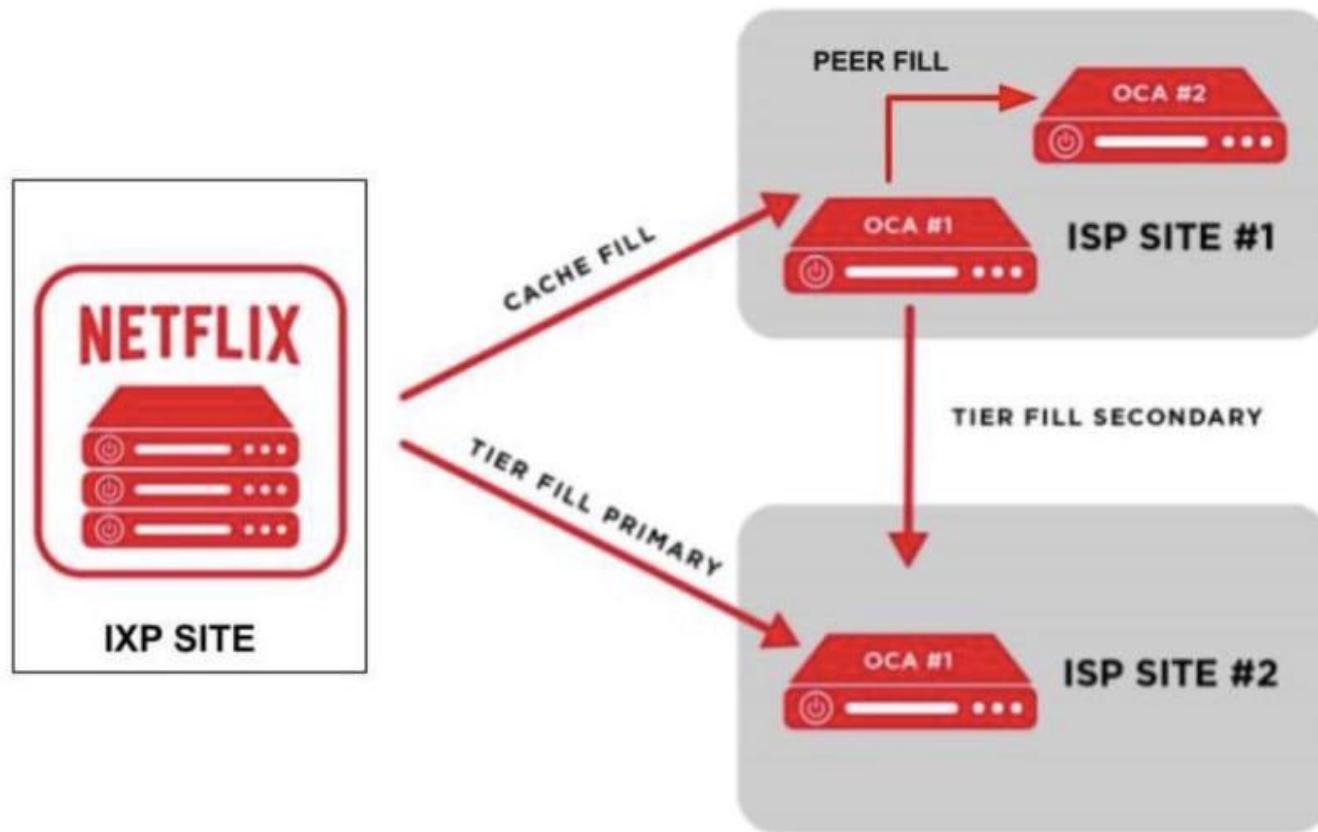


Check exercise
@end of slides-set

Performance effect:

$$E(\text{delay}) = \text{hitRatio} * \text{LocalAccessDelay} + (1 - \text{hitRatio}) * \text{RemoteAccessDelay}$$

Example of caching: Netflix Open Connect



Example Netflix

- Netflix splits video files into small chunks
- Popular videos are stored closed to the end-user
- ISP voluntarily install “Netflix open connect” servers to reduce remote access.
- During off-peak periods, content is moved between the open connect servers

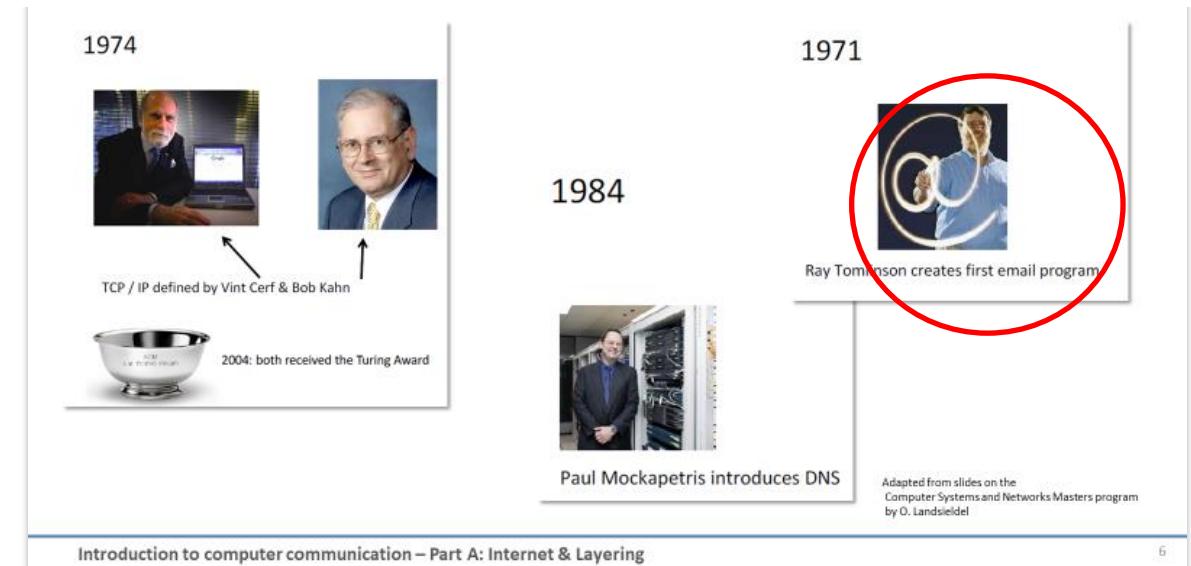
Time for a short break



Image created by Bing Copilot

Roadmap

- Addressing
- Applications needs from transport layer
- Http
 - General description and functionality
 - About maintaining state
 - Performance:
 - Persistence, pipelining
 - Caching and proxies
- **SMTP (POP, IMAP)**
- DNS



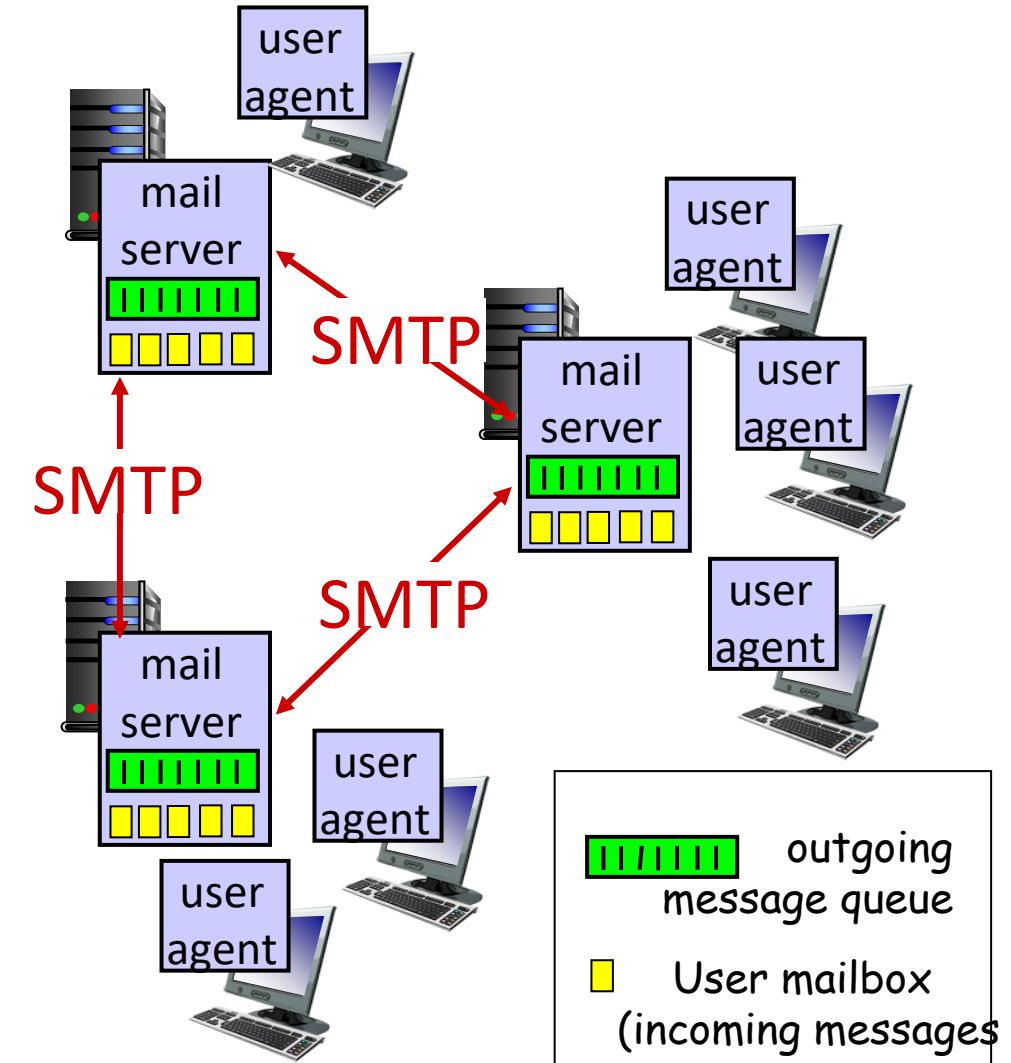
Electronic Mail

User Agent

- a.k.a. “mail reader”: composing, editing, reading mail messages -e.g., outlook, iPhone mail client

Mail Servers

- Simple Mail Transfer Protocol (SMTP) between mail servers to send email messages
 - “client”: sending mail server
 - server: receiving mail server



Scenario: Alice sends e-mail to Bob

1) Alice uses UA to compose e-mail message “to” bob@someschool.edu

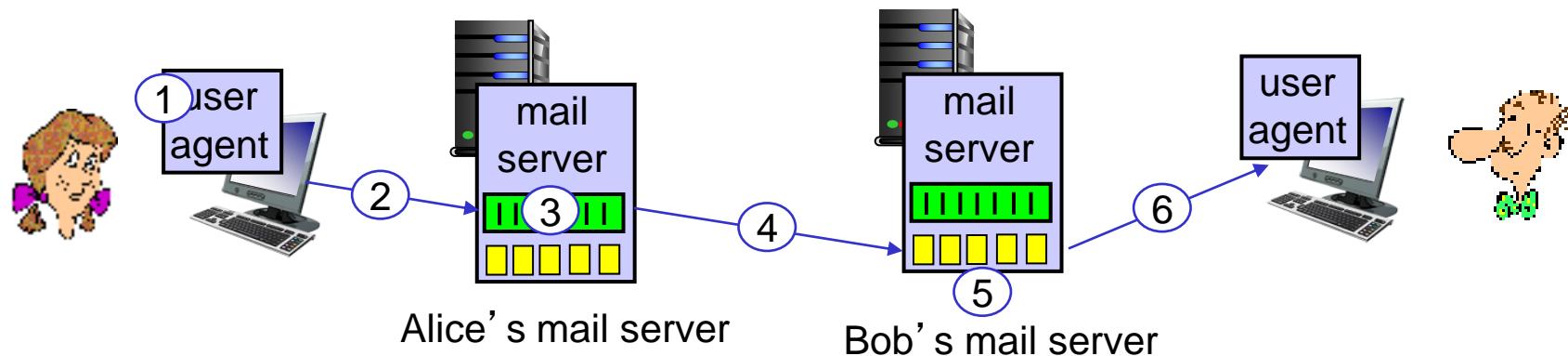
2) Alice’s UA sends message to her mail server; message placed in message queue

3) client side of SMTP opens TCP connection with Bob’s mail server

4) SMTP client sends Alice’s message over the TCP connection

5) Bob’s mail server places the message in Bob’s mailbox

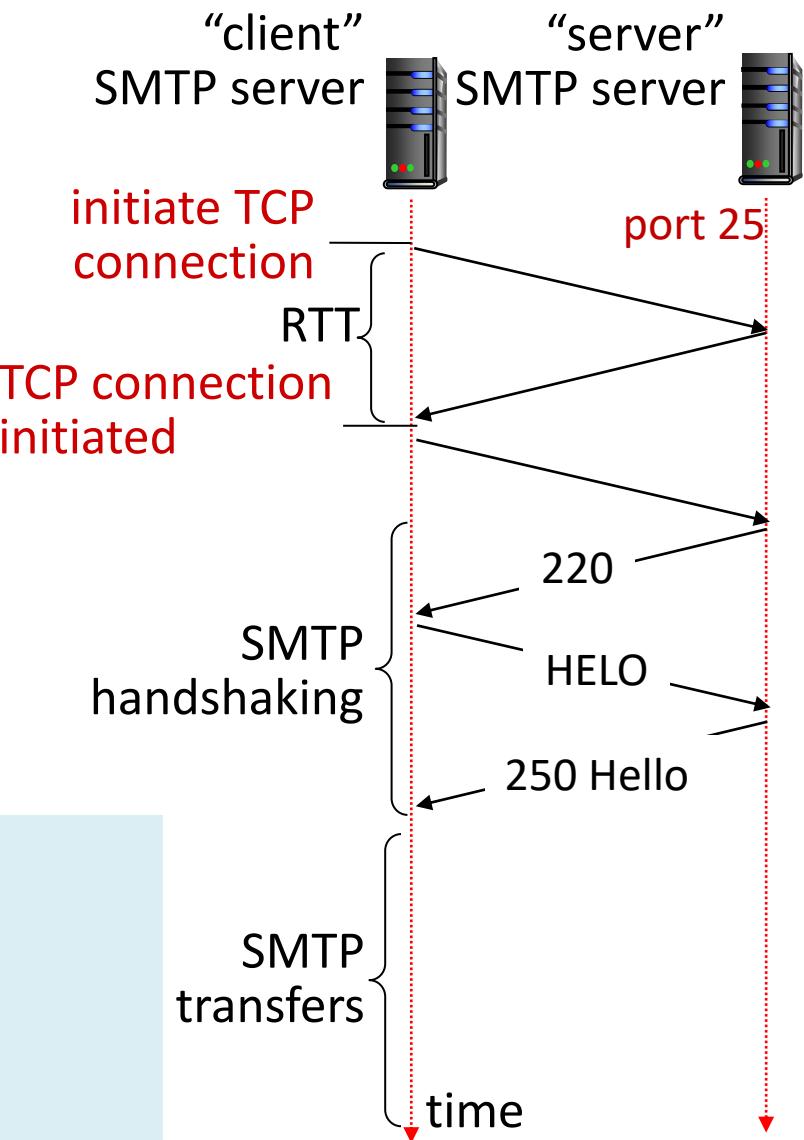
6) Bob invokes his user agent to read message



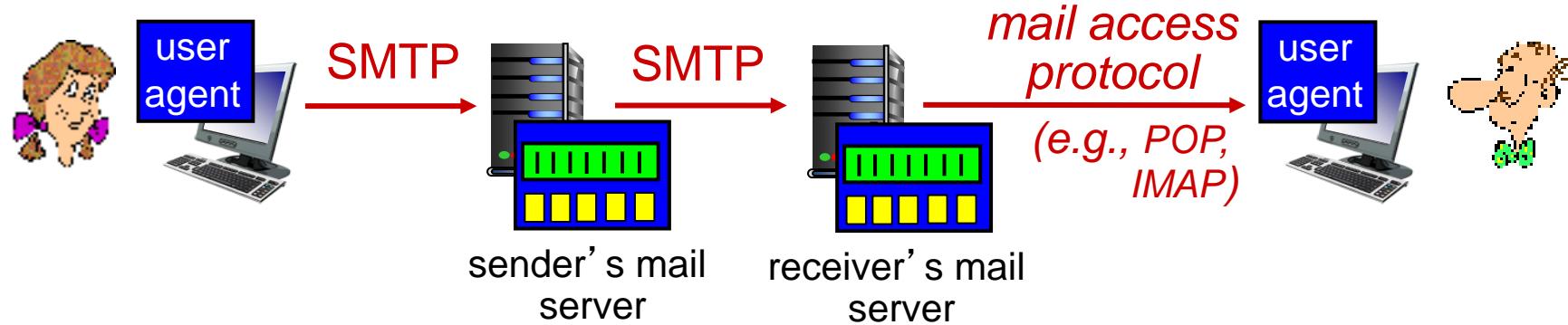
Sample SMTP interaction

```
S: 220 hamburger.edu
C: HELO crepes.fr
S: 250 Hello crepes.fr, pleased to meet you
C: MAIL FROM: <alice@crepes.fr>
S: 250 alice@crepes.fr... Sender ok
C: RCPT TO: <bob@hamburger.edu>
S: 250 bob@hamburger.edu ... Recipient ok
C: DATA
S: 354 Enter mail, end with "." on a line by itself
C: Do you like ketchup?
C: How about pickles?
C: .
S: 250 Message accepted for delivery
C: QUIT
S: 221 hamburger.edu closing connection
```

- SMTP (RFC 2821, 5321), ASCII interface (as basic http)
- three phases
 - handshaking (greeting)
 - transfer of messages
 - closure



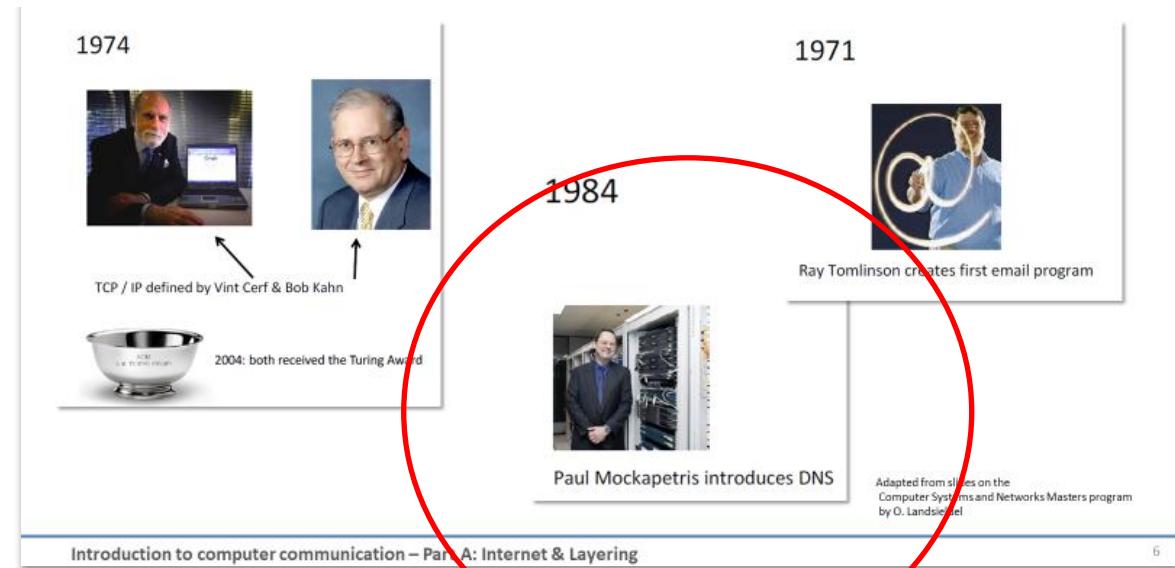
Mail access protocols



- **SMTP:** delivery/storage to receiver' s server
- mail access protocol: retrieval from server
 - **POP:** Post Office Protocol [RFC 1939]: authorization, download
 - **IMAP:** Internet Mail Access Protocol [RFC 1730]: more features, including manipulation of stored msgs on server

Roadmap

- Addressing
- Applications needs from transport layer
- Http
 - General description and functionality
 - About maintaining state
 - Performance:
 - Persistence, pipelining
 - Caching and proxies
- SMTP (POP, IMAP)
- DNS



DNS: Domain Name System

People: many identifiers:

- Name, Passport #, Personnummer

Internet hosts, routers: IP address, 32 bit (IPv4) or 128bit (IPv6), - used for addressing datagrams

- IPv4: 129.16.237.85
- IPv6: 2001:db8:0:1234:0:567:8:1

- Binary number: easier (efficient) process @ router
- “name”, e.g., (www.cse.chalmers.se): easier for humans

Q: map between IP addresses and name ?

Hostname to IP address translation

- Example: `www.chalmers.se 129.16.71.10`
- File with mapping (may be edited) on the system
 - eg Unix: `/etc/hosts` - Windows: `c:\windows\system32\drivers\etc\hosts`

Does not scale for all possible hosts, hard to change

- All hosts would need one copy of the file
- Impossible on the Internet

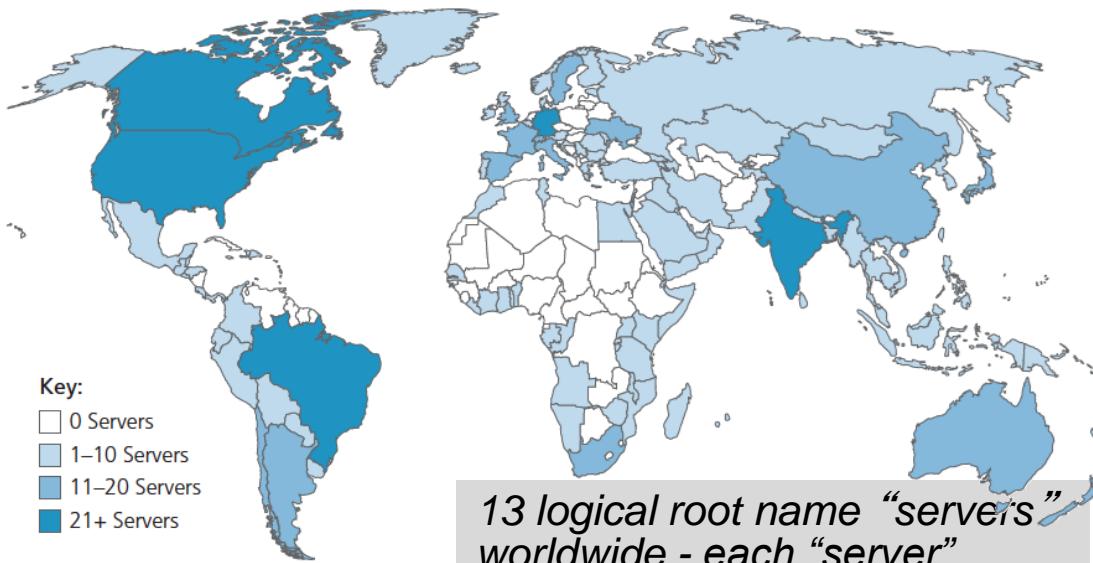
Alternative: DNS, a large **distributed** database

DNS – Domain Name System

why not centralize DNS?

- single point of failure
- traffic volume (many billions of DNS queries/day!)
- maintenance

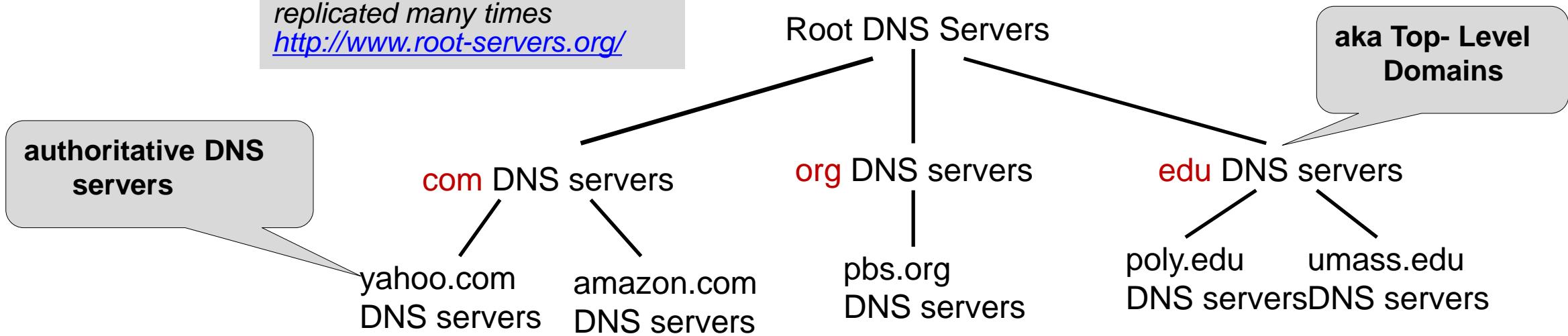
DNS: a distributed, hierarchical database



13 logical root name “servers”
worldwide - each “server”
replicated many times
<http://www.root-servers.org/>

Root DNS servers: incredibly important

- official, contact-of-last-resort: Internet couldn't function without it!
- DNSSEC – extension; provides security (authentication and message integrity)
- Managed by ICANN (Internet Corporation for Assigned Names and Numbers) who manages Internet Assigned Numbers Authority (IANA)



DNS name resolution example

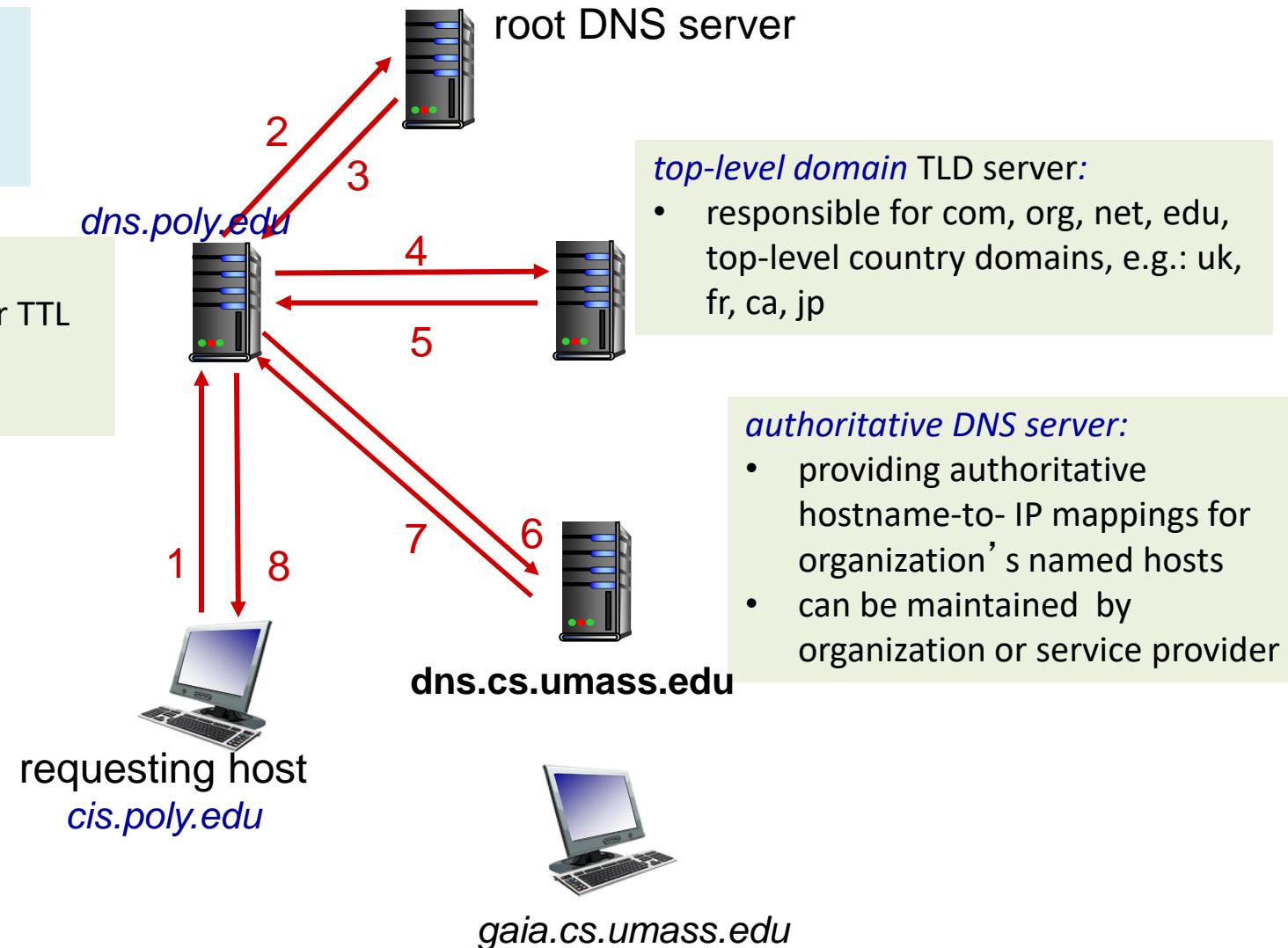
- host at cis.poly.edu wants IP address for gaia.cs.umass.edu

Local name server (aka default name server)

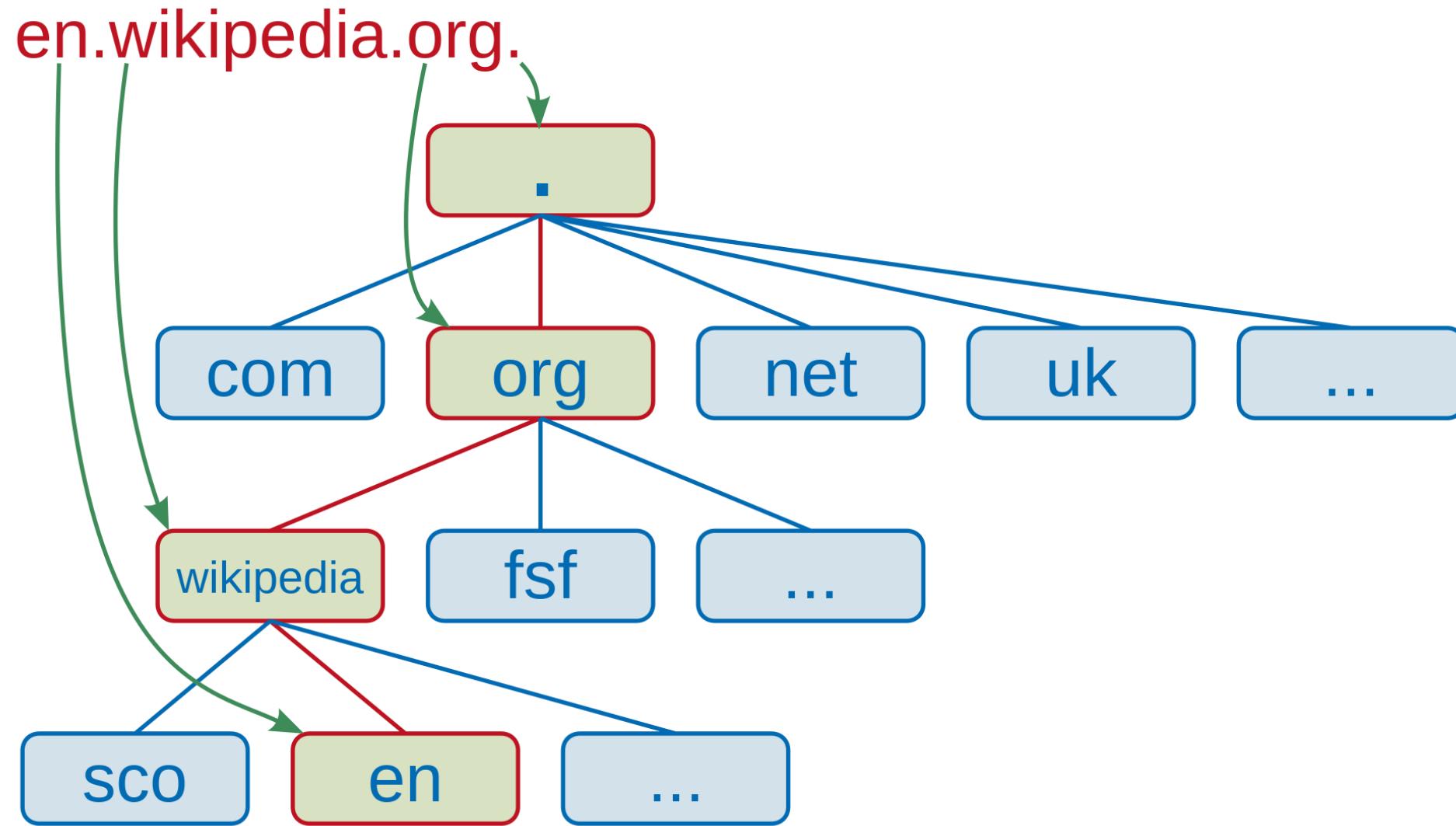
- acts as proxy for clients, caches entries for TTL
- sends queries to DNS hierarchy
- each ISP has one

iterated query:

- contacted server replies with name of server to contact
- “I don’t know this name, but ask this server”



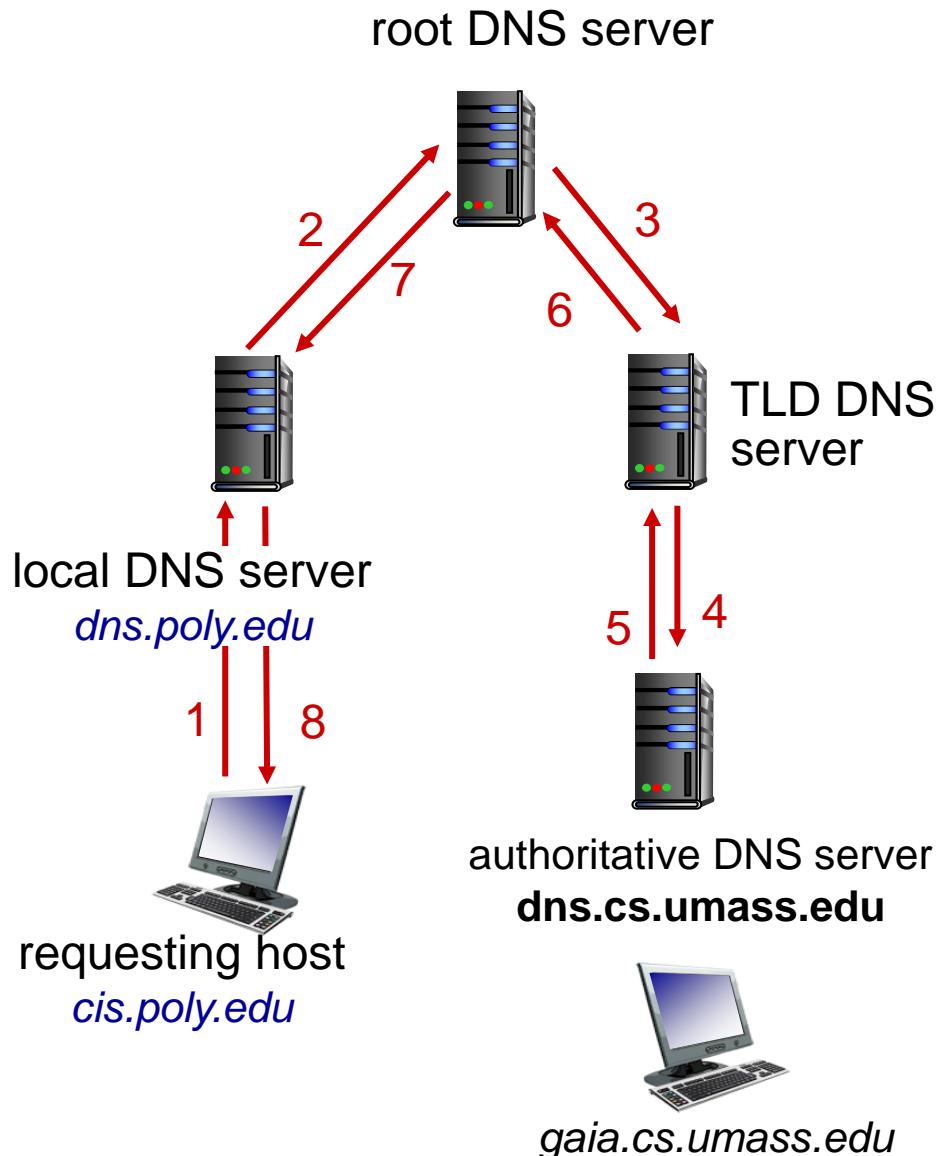
The hierarchy of labels



DNS name resolution, another example

recursive query:

- puts burden of name resolution on contacted name server
- heavy load at upper levels of hierarchy?



DNS services and records

- hostname to IP address translation
- host aliasing: canonical, alias names
- Info for authoritative name DNS server, mail server
- load distribution
 - replicated Web servers: many IP addresses correspond to one name

Resource Record format:
(name, value, type, ttl)

type=A (AAAA for IPv6)

- **name** is hostname
- **value** is IP address

type=NS

- **name** is domain (e.g., foo.com)
- **value** is hostname of authoritative name server for this domain

type=CNAME

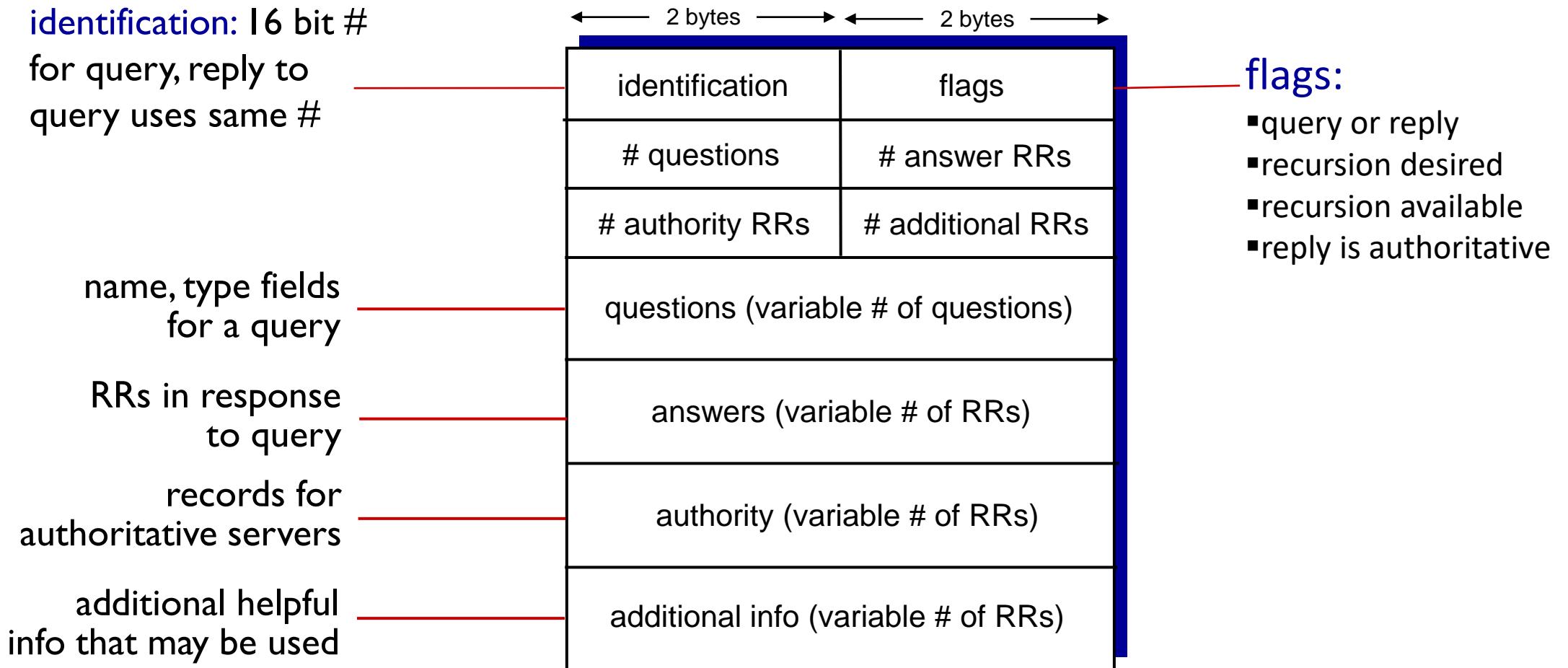
- **name** is alias for the “canonical” (real) name
- **value** is canonical name
- Eg `www.example.com` is really `servereast.backup2.example.com`

type=MX

- **value** is name of mailserver associated with **name**

DNS protocol, messages

- *query* and *reply* messages (use UDP), both with same *message format*
- ASCII (this one too), ie possible to try it -- use the nslookup command)



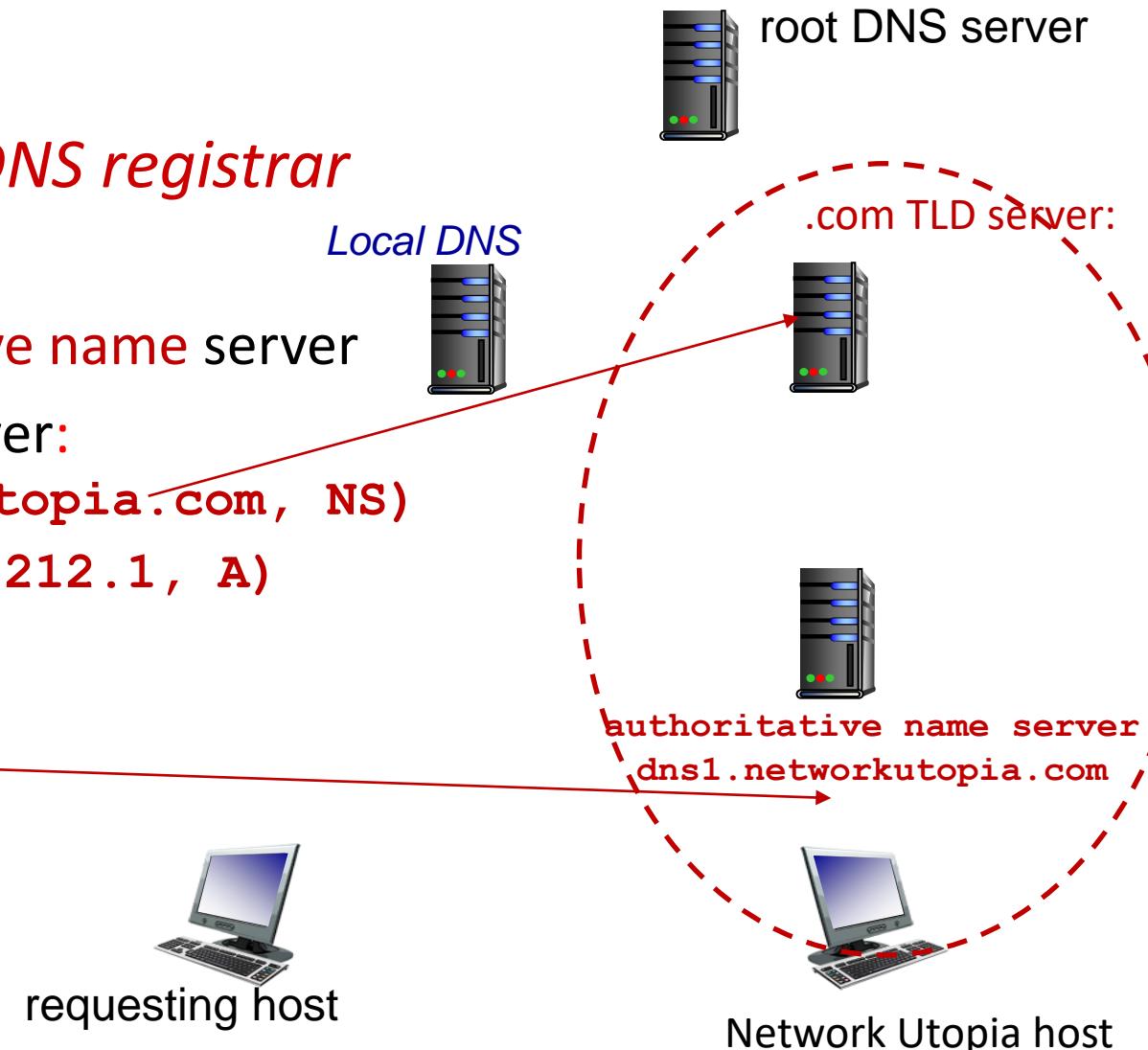
DNS: side note on caching / updating records

- once (any) name server learns mapping, it *caches* it
 - cache timeout after some time (TTL)
- cached entries may be *out-of-date* (best effort name-to-address translation!)
 - if name host changes IP address, may not be known Internet-wide until all TTLs expire
- update/notify mechanisms IETF standard
 - RFC 2136

Inserting records into DNS

Example: new startup “Network Utopia”

- register name networkuptopia.com at *DNS registrar* (e.g., Network Solutions)
 - provide names, IP addresses of **authoritative name server**
 - registrar inserts two RRs into .com TLD server:
 - (`networkutopia.com`, `dns1.networkutopia.com`, NS)
 - (`dns1.networkutopia.com`, `212.212.212.1`, A)
- Adding a **new host/service to domain:**
 - Add to **authoritative name server**
 - type A record for `www.networkuptopia.com`
 - type MX record for `networkutopia.com` (mail)



DNS and security risks

DDoS attacks

- Bombard root/TLD servers
 - Mitigation: local DNS servers cache IPs of TLD/authoritative servers, allowing root/TLD server bypass

DNS extensions include authentication

Redirect attacks

- Man-in-middle
 - Intercept queries
- DNS poisoning
 - Send bogus replies to DNS server, which it caches

Exploit DNS for DDoS

- Send queries with spoofed source address

DNSSEC
[RFC 4033]

Summary

- Addressing
- Applications' needs from Transport layer
- Application architectures
 - client-server
 - *(p2p: will study later in the course, after the layers-centered study)*
- *Representative protocols:*
 - HTTP (incl. caching etc)
 - SMTP (POP, IMAP)
 - DNS



More application-layer protocols coming soon, after a pass of the 4 top layers
- P2P applications
- video streaming, content distribution networks (CDN) and more

Resources

("Part #", as in the lectures modules)	8th Edition, careful study	8th Edition, quick reading	7th Edition, careful study	7th Edition, quick reading
Part 2: App-layer, basic protocols	Sections 2.2-2.4	Sections 2.1 and 2.7	the same + Section on http2e	the same
Part 2: Application Layer	Chapter 2 Exercises 7, 8, 15, and 21 <i>Additional nice-to-solve: Chapter 2, Exercises 1, 3, 19, 20</i>		Chapter 2 Exercises 7, 8, 15, and 21 (Note: 7 and 8 are slightly different but on the same topic) <i>Additional nice-to-solve: the same</i>	

Thanks for joining!

Extra Material

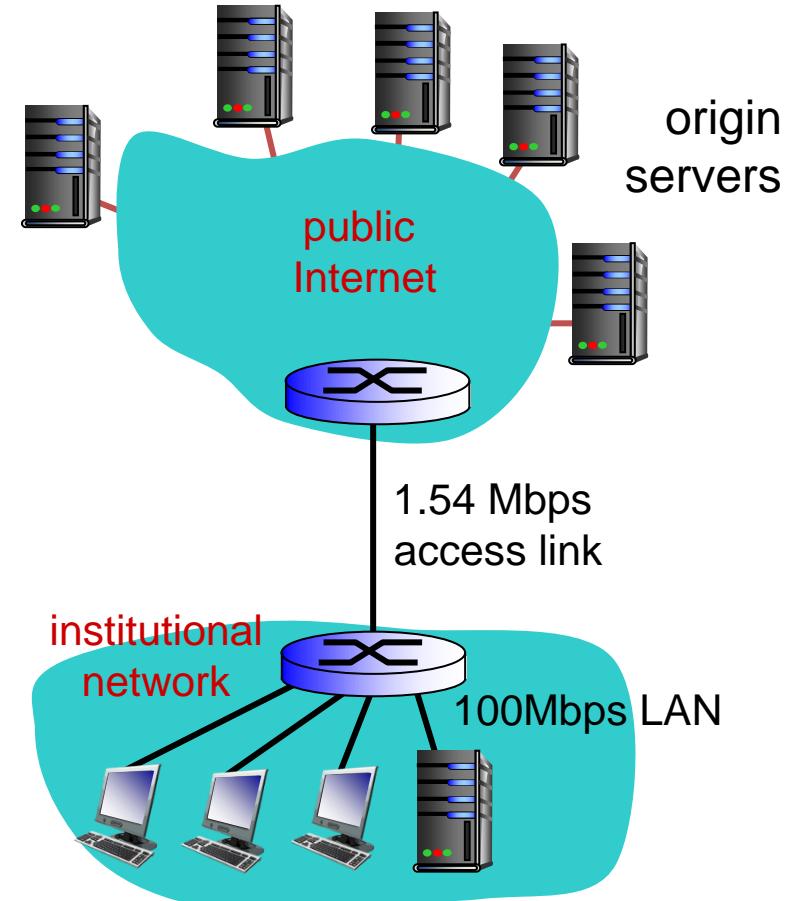
Caching example, as an exercise:

assumptions:

- ❖ avg object size: 100K bits
- ❖ avg request rate from browsers to origin servers: 15/sec
 - ❖ i.e., avg data rate to browsers: 1.50 Mbps
- ❖ RTT (round-trip-time) from institutional router to any origin server: 2 sec
- ❖ access link rate: 1.54 Mbps

consequences:

- ❖ LAN utilization: 1.5% *problem!*
- ❖ access link utilization = **99%**
- ❖ total delay = Internet delay + access delay + LAN delay
= 2 sec + minutes + quite_small



Caching example: faster access link

assumptions:

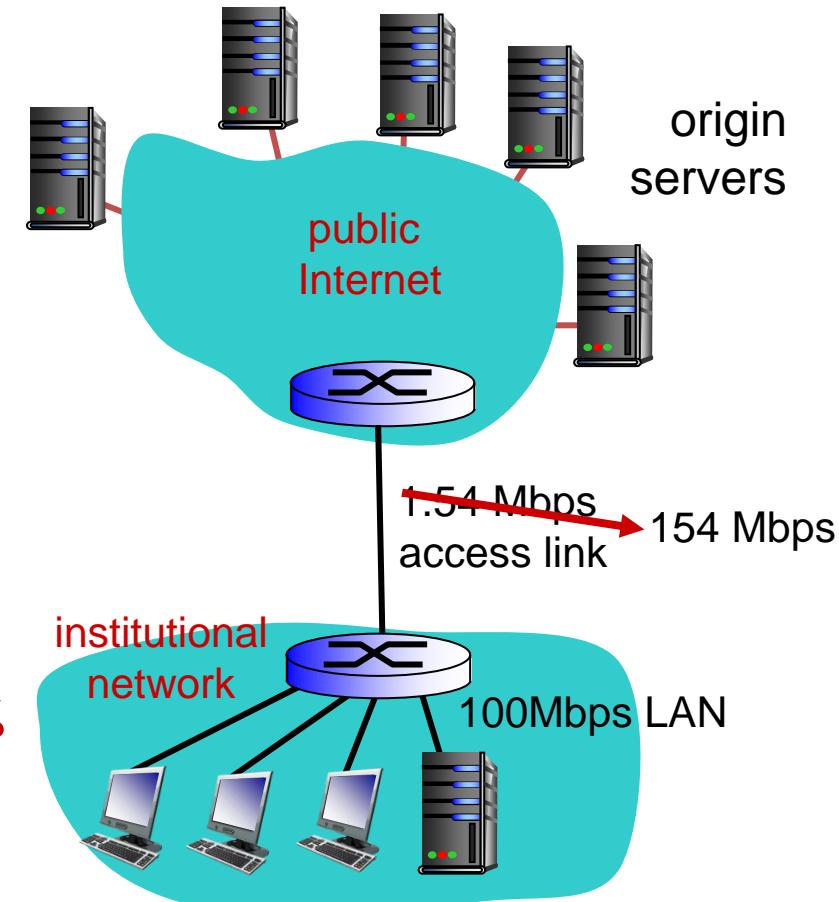
- ❖ avg object size: 100K bits
- ❖ avg request rate from browsers to origin servers: 15/sec
 - ❖ i.e., avg data rate to browsers: 1.50 Mbps
- ❖ RTT from institutional router to any origin server: 2 sec
- ❖ access link rate: ~~1.54 Mbps~~

consequences:

~~154 Mbps~~

- ❖ LAN utilization: 1.5%
- ❖ access link utilization = ~~99%~~ \rightarrow 9.9%
- ❖ total delay = Internet delay +
access delay + LAN delay
 $= 2 \text{ sec} + \text{minutes} + \text{usecs}$

~~msecs~~



Cost: increased access link speed (not cheap!)

Caching example: install local cache

assumptions:

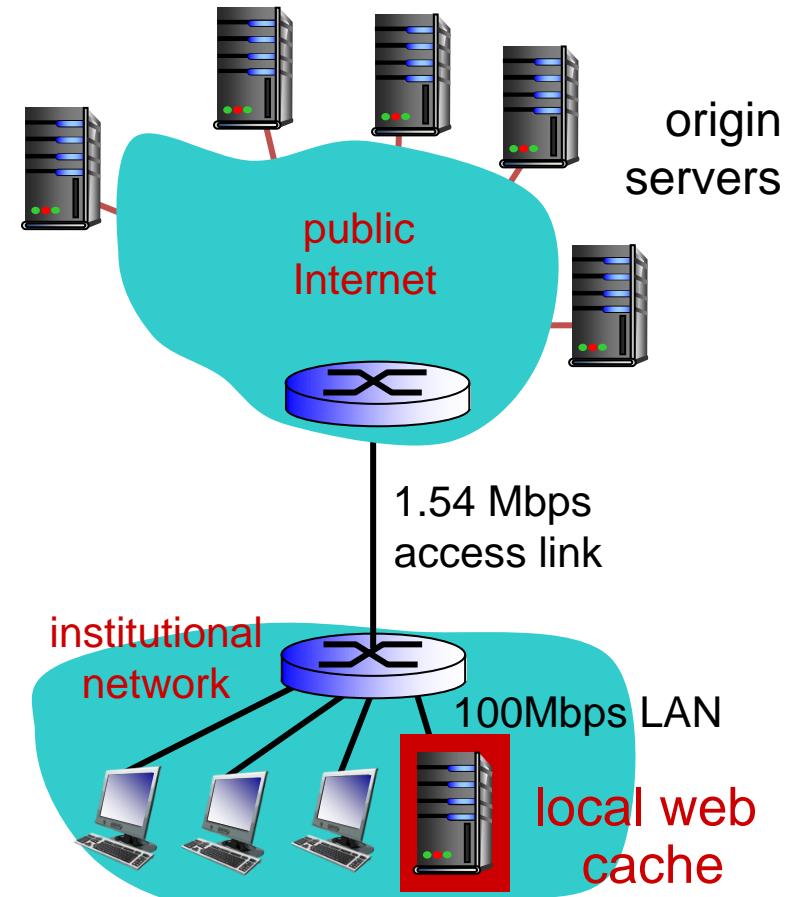
- ❖ avg object size: 100K bits
- ❖ avg request rate from browsers to origin servers: 15/sec
 - ❖ i.e., avg data rate to browsers: 1.50 Mbps
- ❖ RTT from institutional router to any origin server: 2 sec
- ❖ access link rate: 1.54 Mbps

consequences:

- ❖ LAN utilization: 1.5%
- ❖ access link utilization ?
- ❖ total delay ?

How to compute link utilization, delay?

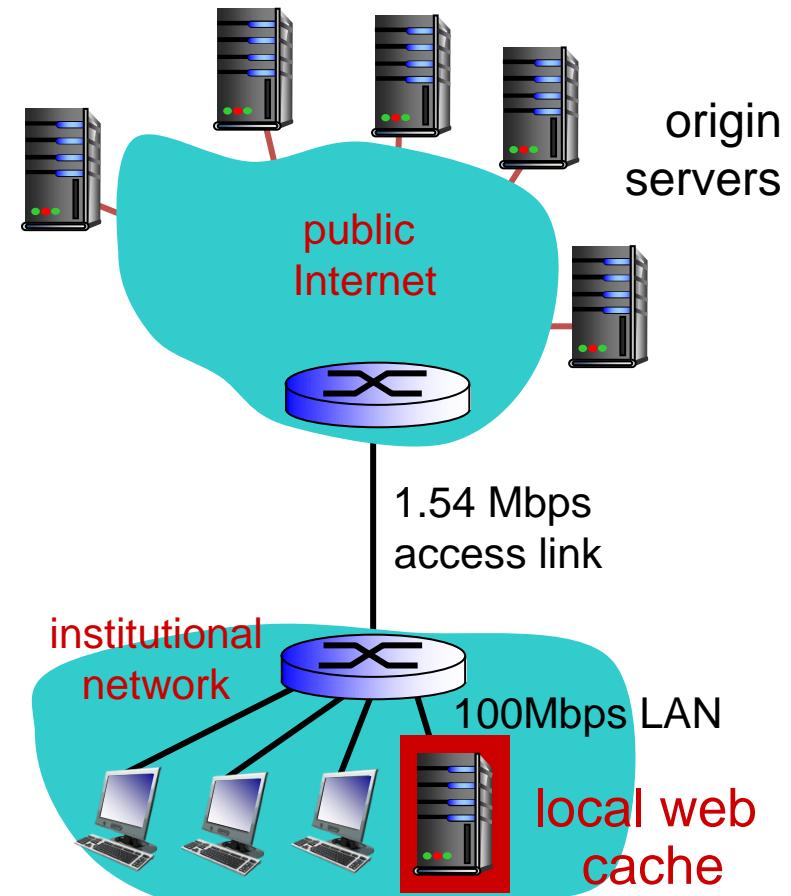
Cost: web cache (cheap!)



Caching example: install local cache

*Calculating access link utilization,
delay with cache:*

- suppose cache hit rate is 0.4
 - 40% requests satisfied at cache, 60% requests satisfied at origin
- ❖ access link utilization:
 - 60% of requests use access link
- ❖ data rate to browsers over access link
 $= 0.6 * 1.50 \text{ Mbps} = .9 \text{ Mbps}$
 - utilization $= 0.9 / 1.54 = .58 = 58\%$
- ❖ total delay
 - $= 0.6 * (\text{delay from origin servers}) + 0.4 * (\text{delay when satisfied at cache})$
 - $= 0.6 (2 \text{ s}) + 0.4 (\sim \text{msecs})$
 - $= \sim 1.2 \text{ secs}$
 - less than with 154 Mbps link (and cheaper too!)



Review question

Client-Server vs Peer 2 Peer: Which of the characteristics below are associated with a client-server approach to structuring network applications (as opposed to a P2P approach)?

There is a server that is always on.

A process requests service from those it contacts and will provide service to processes that contact it.

HTTP uses this application structure.

There is a server with a well known server IP address.

There is not a server that is always on.

Client-Server
p2p

Review question

COOKIES. What is an **HTTP cookie** used for?

1. A cookie is a code used by a client to authenticate a person's identity to an HTTP server.
2. A cookie is a code used by a server, carried on a client's HTTP request, to access information the server had earlier stored about an earlier interaction with this person. [Think about the distinction between a browser and a person.]
3. Like dessert, cookies are used at the end of a transaction, to indicate the end of the transaction.
4. A cookie is used to spoof client identity to an HTTP server.
5. A cookie is a code used by a server, carried on a client's HTTP request, to access information the server had earlier stored about an earlier interaction with this Web browser. [Think about the distinction between a browser and a person.]

Review question

A DETAILED LOOK AT AN HTTP GET (3) suppose a client is sending an HTTP GET request message to a web server, gaia.cs.umass.edu. Suppose the client-to-server HTTP GET message is the following (same as in previous problem):

```
GET /kurose_ross_sandbox/interactive/quotation2.htm HTTP/1.1
Host: gaia.cs.umass.edu
Accept: text/plain, text/html, text/xml, image/jpeg, image/gif, audio/mpeg,
audio/mp4, video/wmv, video/mp4,
Accept-Language: en-us, en-gb;q=0.1, en;q=0.7, fr, fr-ch, da, de, fi
If-Modified-Since: Wed, 09 Sep 2020 16:06:01 -0700
User Agent: Mozilla/5.0 (Windows NT 6.1; WOW64) AppleWebKit/535.11 (KHTML, like
Gecko) Chrome/17.0.963.56 Safari/535.11
```

Does the client have a cached copy of the object being requested?

1. Yes, because this is a conditional GET, as evidenced by the If-Modified-Since field.
2. No, because a client would not request an object if it had that object in its cache.
3. Yes, because HTTP 1.1 is being used.
4. There's not enough information in the header to answer this question.

Review Question

DNS CACHING. What is the value of caching in the local DNS name server? Check all that apply.

1. DNS caching provides the ability to serve as authoritative name server for multiple organizations.
2. DNS caching provides prioritized access to the root servers, since the DNS request is from a local DNS cache.
3. DNS caching results in less load elsewhere in DNS, when the reply to a query is found in the local cache.
4. DNS caching provides for faster replies, if the reply to the query is found in the cache



Course on Computer Communication and Networks

Lecture 4

Chapter 3; Transport Layer

Part A: Reliable Data Transfer and Addressing

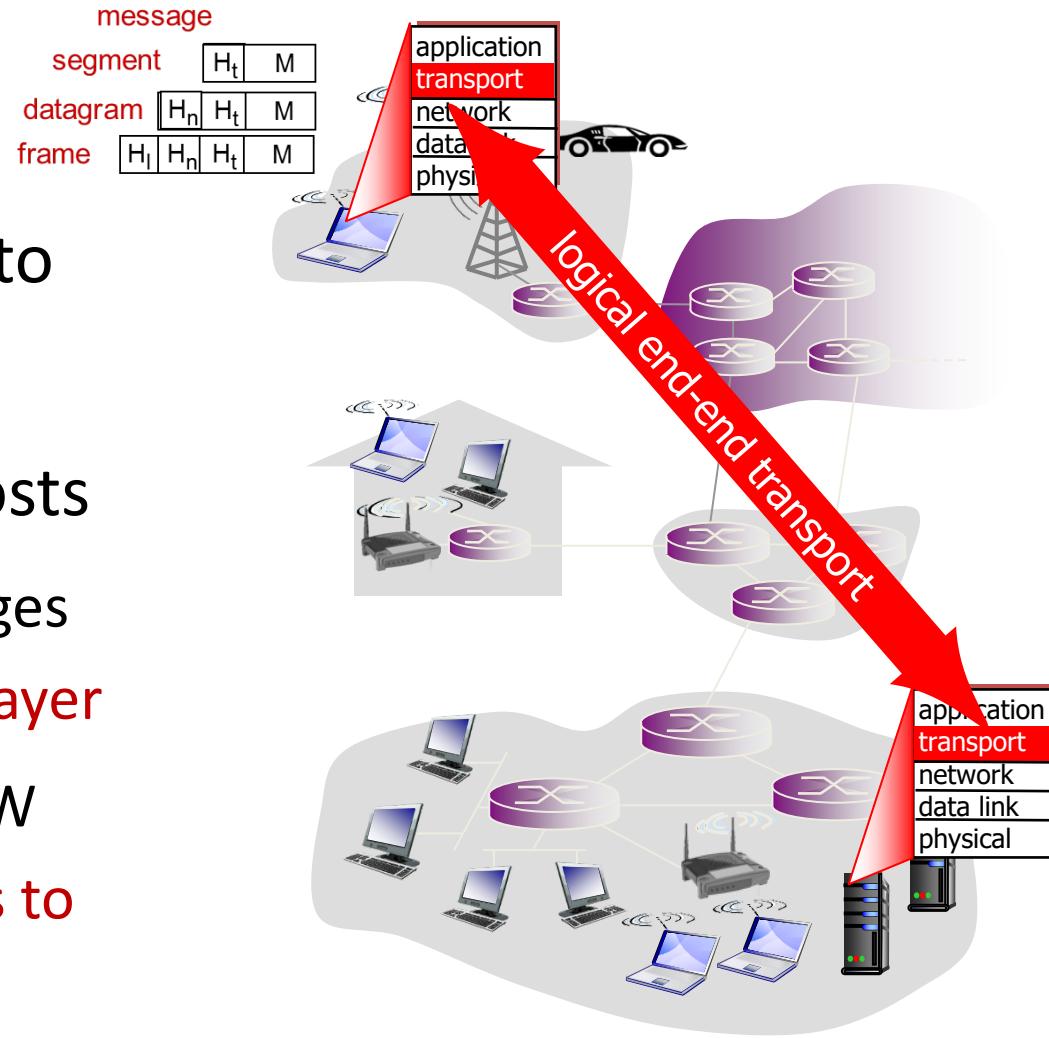
EDA344/DIT 423/ LEU062

Lecturer: M. Papatriantafilou

Based on the book Computer Networking: A Top Down Approach, Jim Kurose, Keith Ross, Addison-Wesley.

Transport services and protocols

- provide communication services to app-layer protocols
- transport protocols run in end-hosts
 - send side: encapsulates app messages into **segments**, passes to network layer
 - rcv side: **receives** segments from NW layer, passes the included messages to app layer



Go to www.menti.com and use the code 6145 9030

About the network layer services of the Internet



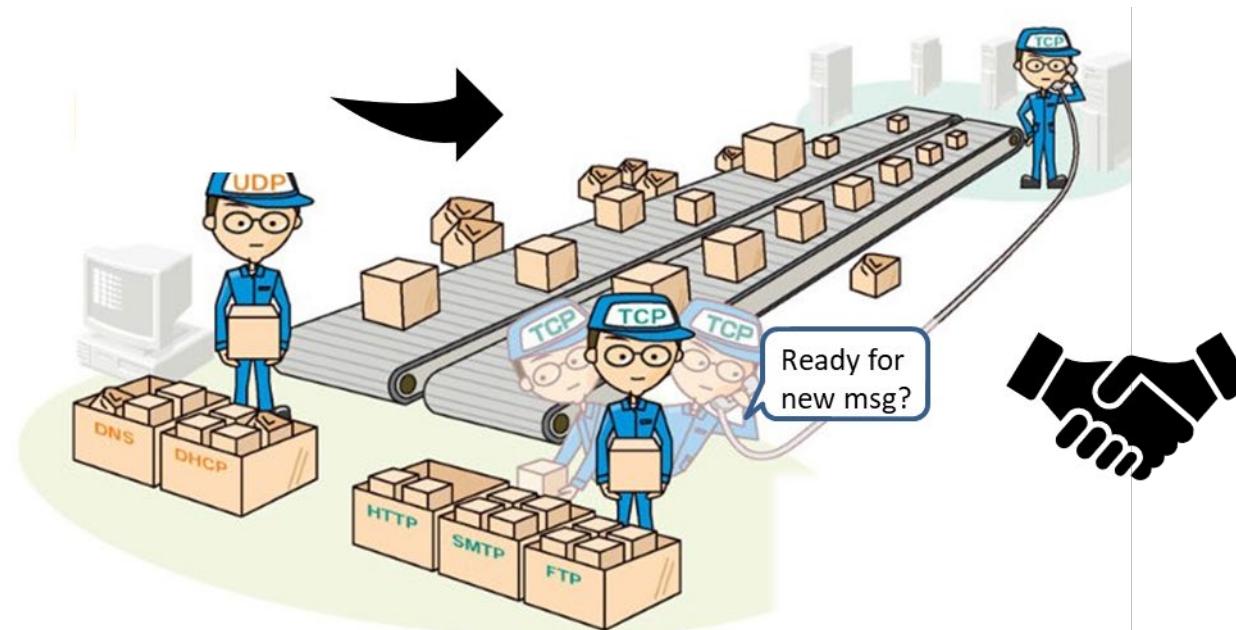
Internet's principal transport-layer protocols

Best effort delivery (might miss, reorder segments; *connectionless service*): **UDP (User Datagram Prot)**

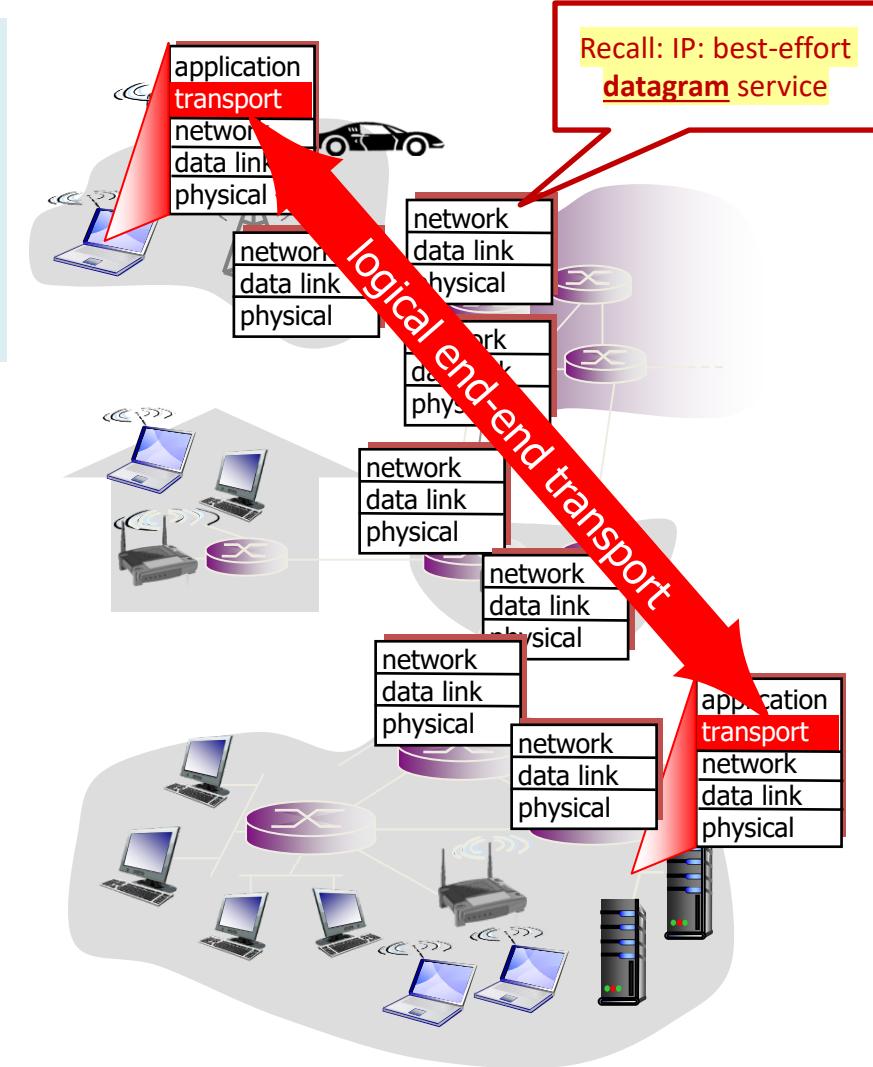
- “careless protocol”: no-frills extension of “best-effort” IP

Reliable, in-order delivery (*connection-oriented service*): **TCP (Transmission Control Prot)**

- “careful protocol”; also provides
 - flow control (care for buffer space at receiver)
 - care for the health of the network (aka TCP’s congestion control)



No time or throughput guarantees
Both support addressing (encapsulation), of course!



Roadmap

Transport Layer: Learning goals:

- understand principles of transport layer services:
 - addressing, multiplexing/demultiplexing
 - reliable data transfer
 - flow control
 - congestion control (not really a pure Transport-layer issue – *first discuss here since TCP has it; more in connection with RealTime traffic*)
- instantiation and implementation in the Internet (TCP, UDP)

- Transport layer services in Internet
- Addressing, multiplexing/demultiplexing
- Principles of reliable data transfer
- Connectionless, unreliable transport: UDP

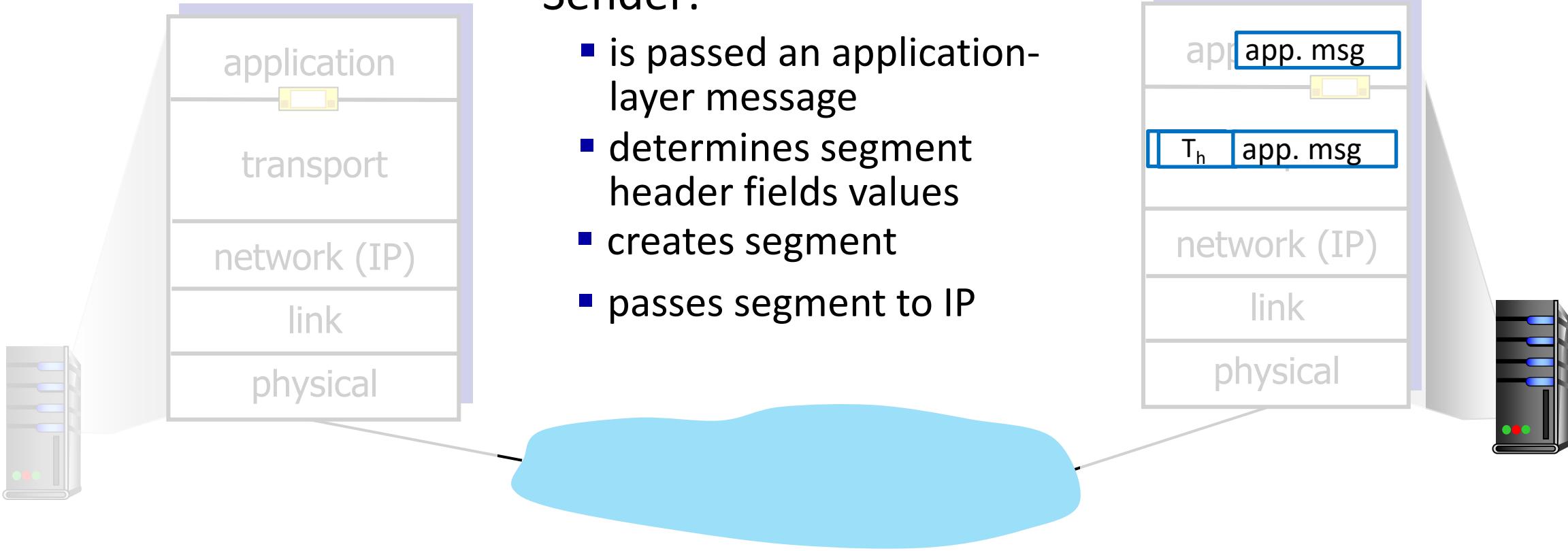
Next lecture: connection-oriented transport: TCP
– *reliable transfer in practice + more stuff TCP does*



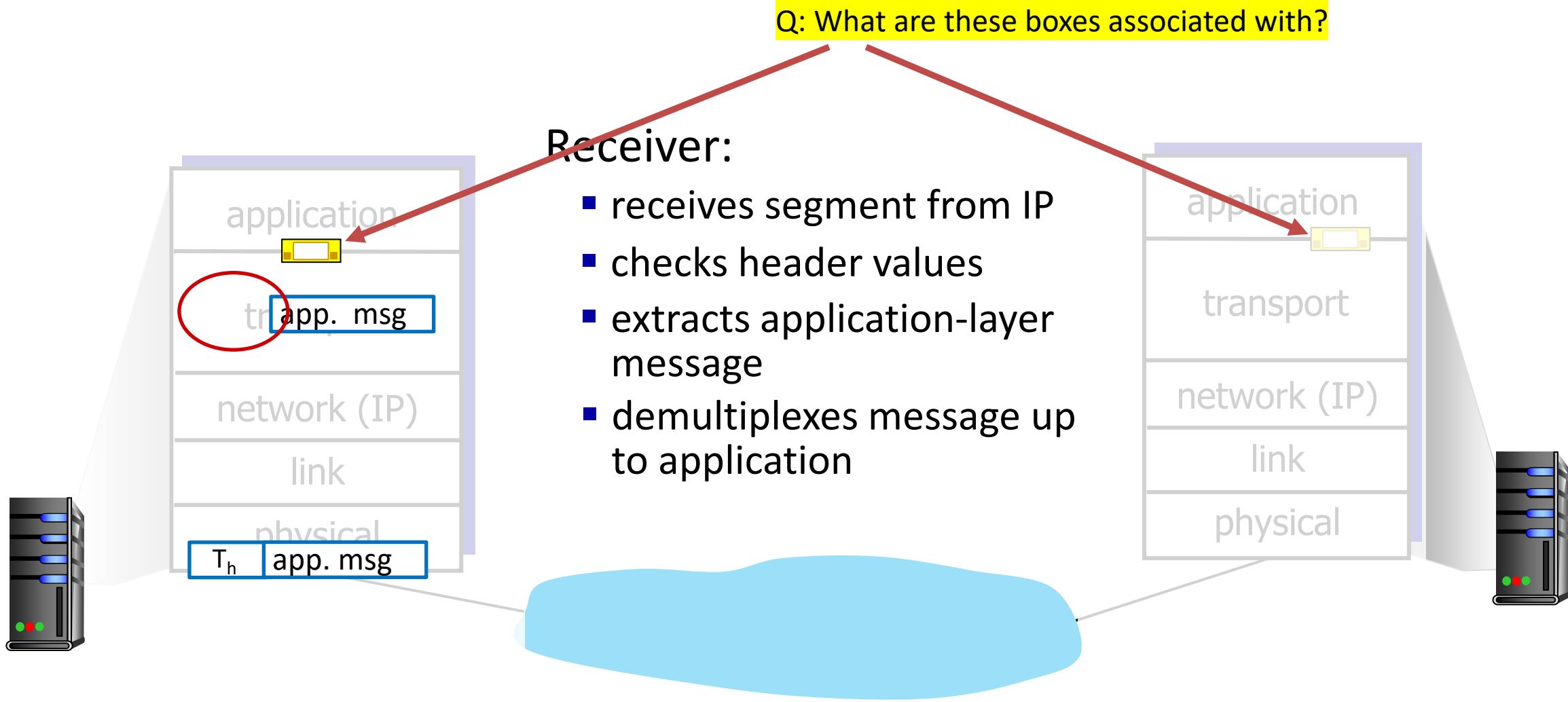
Transport Layer Actions

Sender:

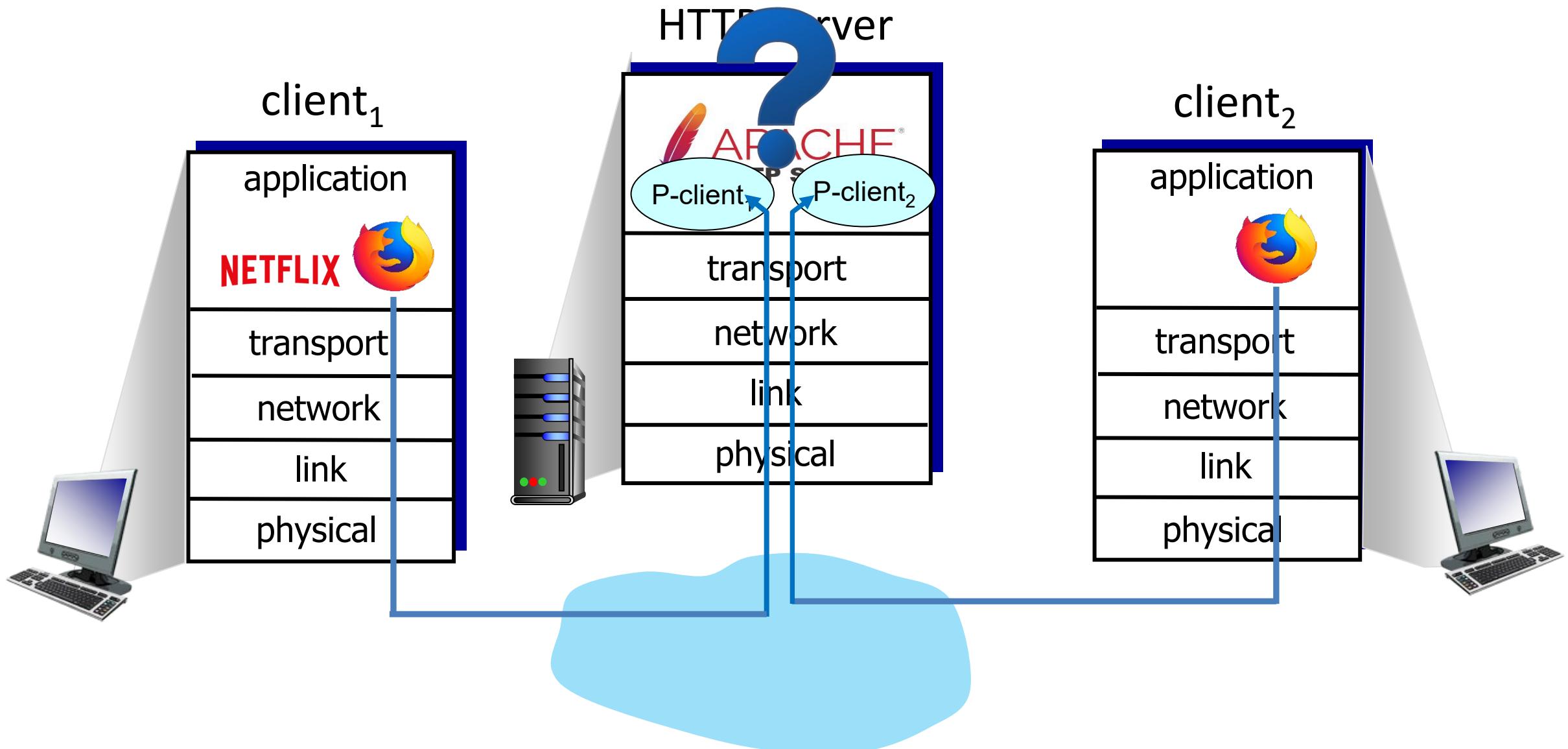
- is passed an application-layer message
- determines segment header fields values
- creates segment
- passes segment to IP



Transport Layer Actions

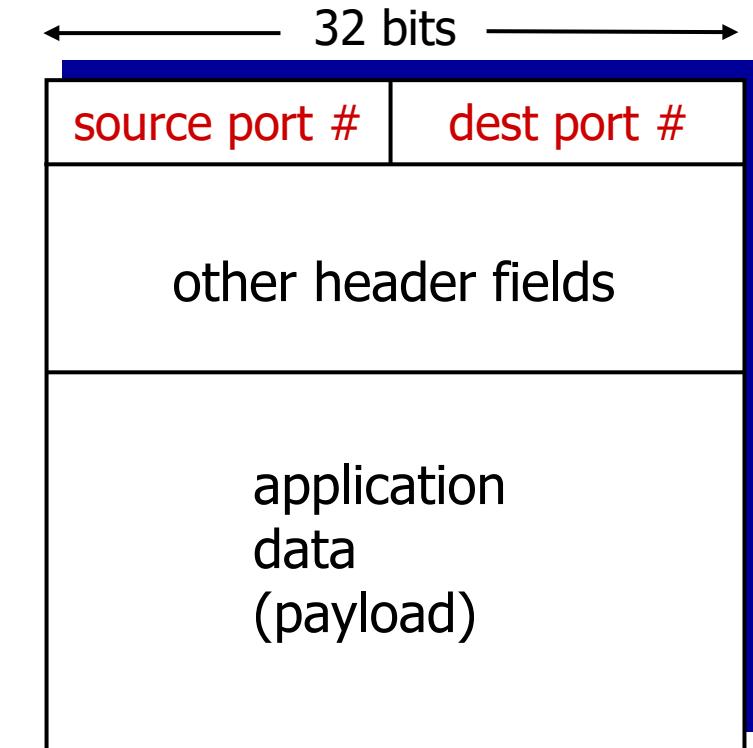
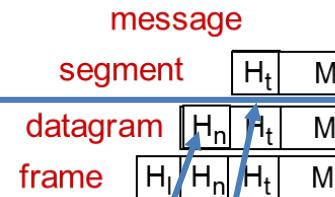


Addressing: Multiplexing/demultiplexing



Addressing (implementation clarification)

- Host receives IP datagrams
 - Datagram (i.e. IP packet) has source, destination IP address
 - datagram carries transport-layer segment
 - segment has source, destination port number
- Host's transport protocols can use *IP addresses & port numbers* to direct segment to appropriate socket



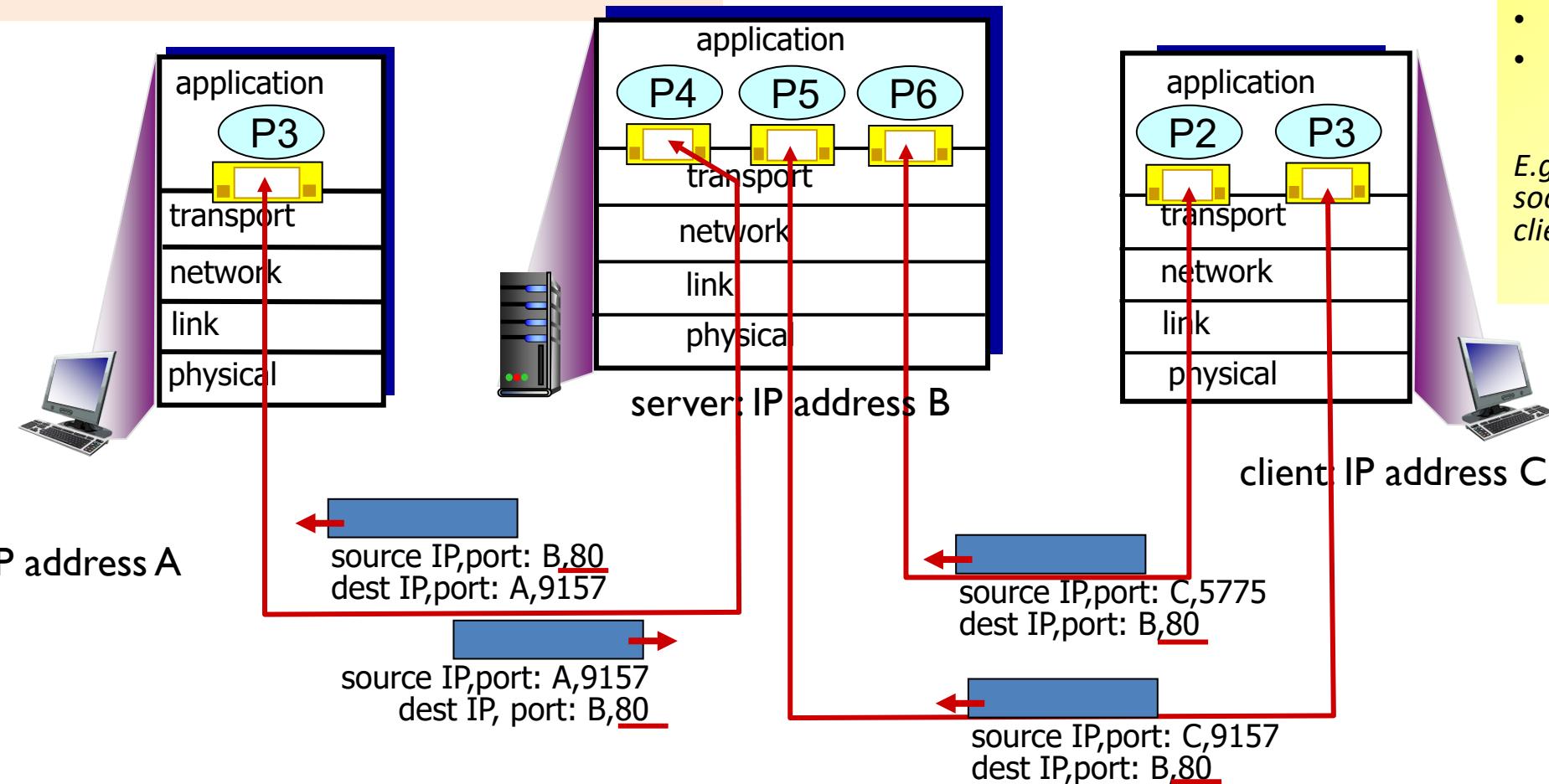
TCP/UDP (abstract) segment format

Connection-oriented (TCP) addressing + example

TCP socket identified by 4-tuple:

(sourceIPaddr, source port#, destIPaddr, dest port#)

(Demultiplexing) Receiver uses **all 4 values** to direct segment to **appropriate socket**

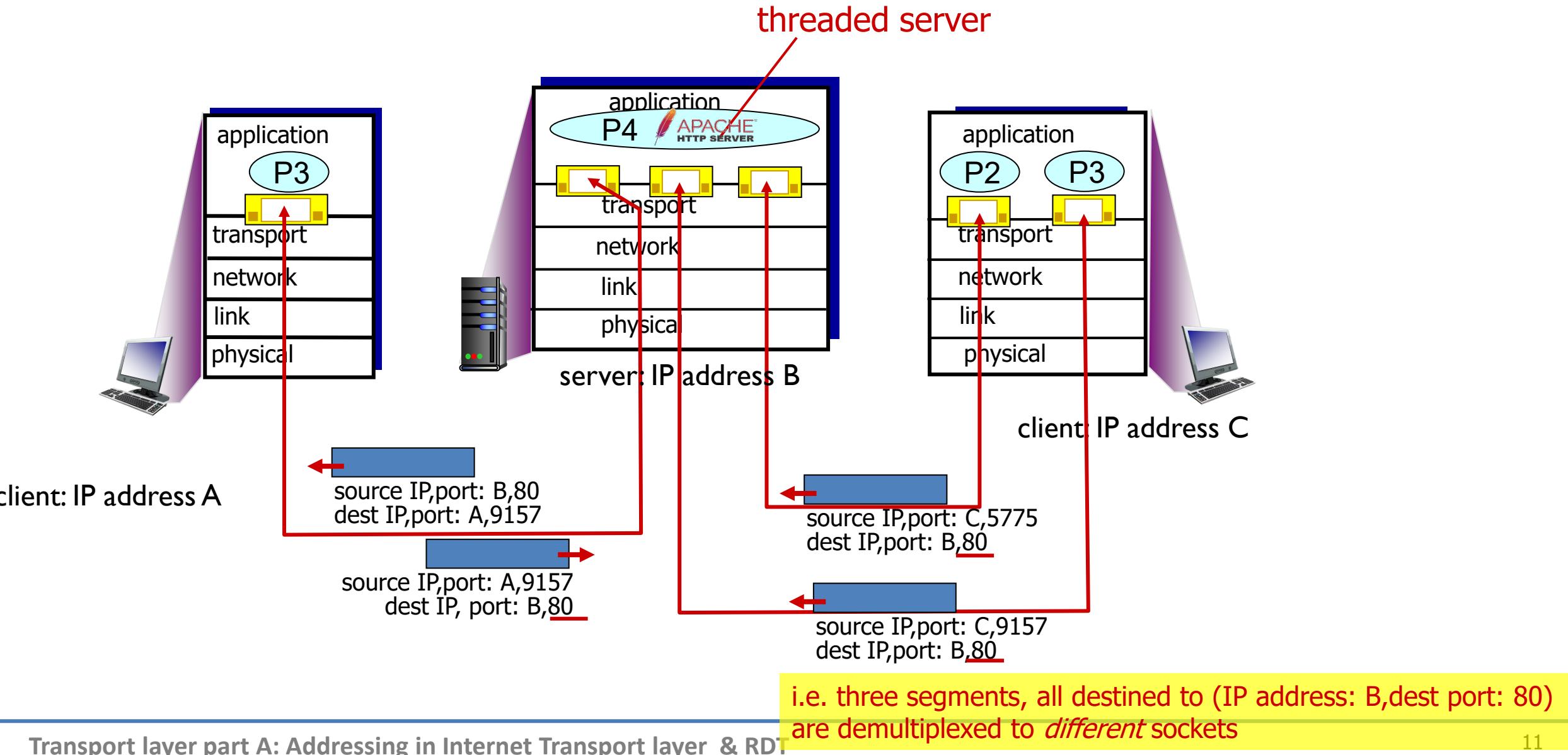


A server may support many simultaneous TCP sockets:

- one socket per connection
- each identified by its own 4-tuple

E.g. web servers have different sockets for each connecting client

TCP demux: Threaded web server

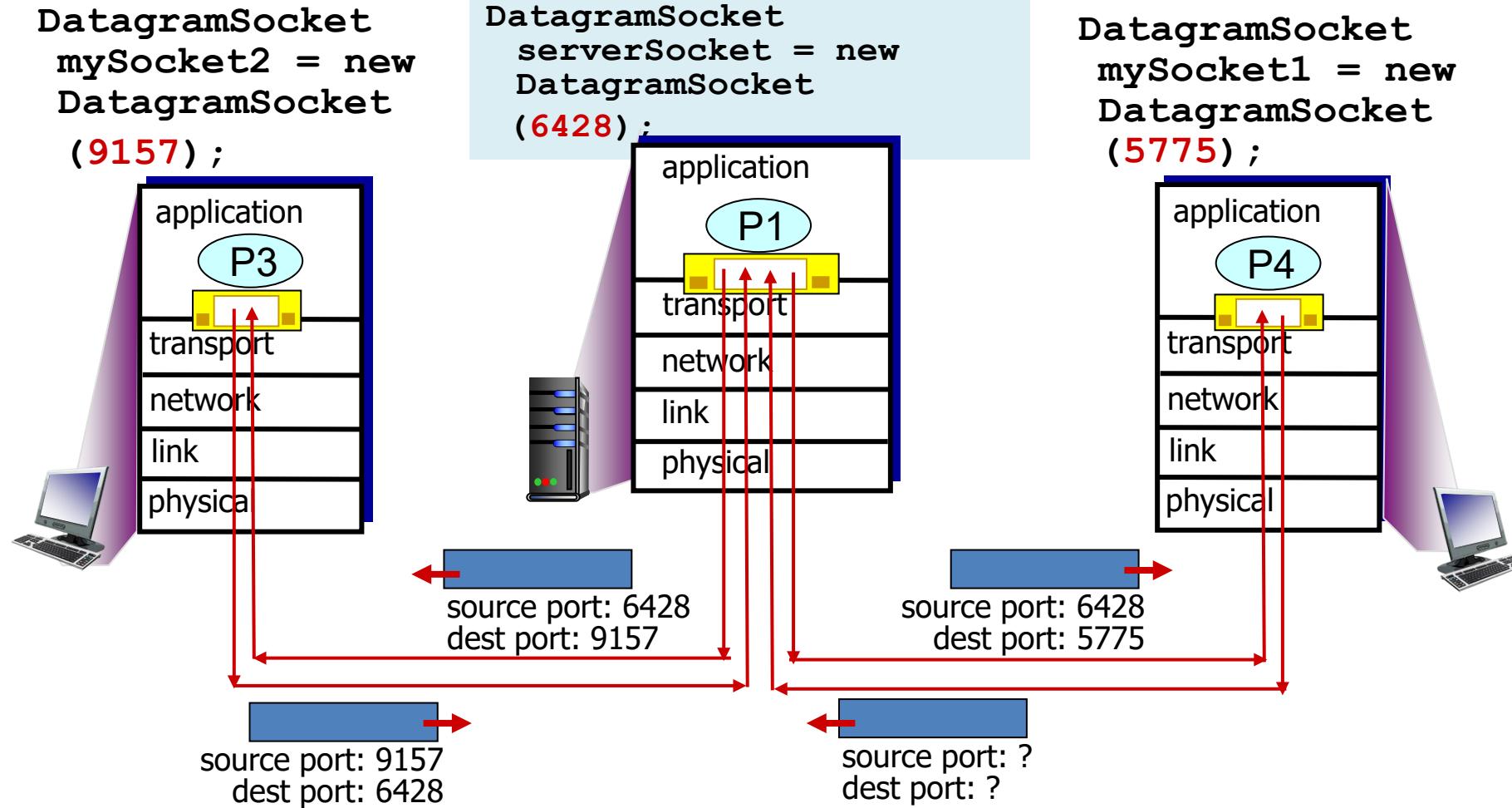


UDP addressing – demultiplexing + example

when host receives UDP segment:

- directs UDP segment to socket with that port #

segments with *same destination port#* (but perhaps different source IP addresses or source port#) go to *the same socket*



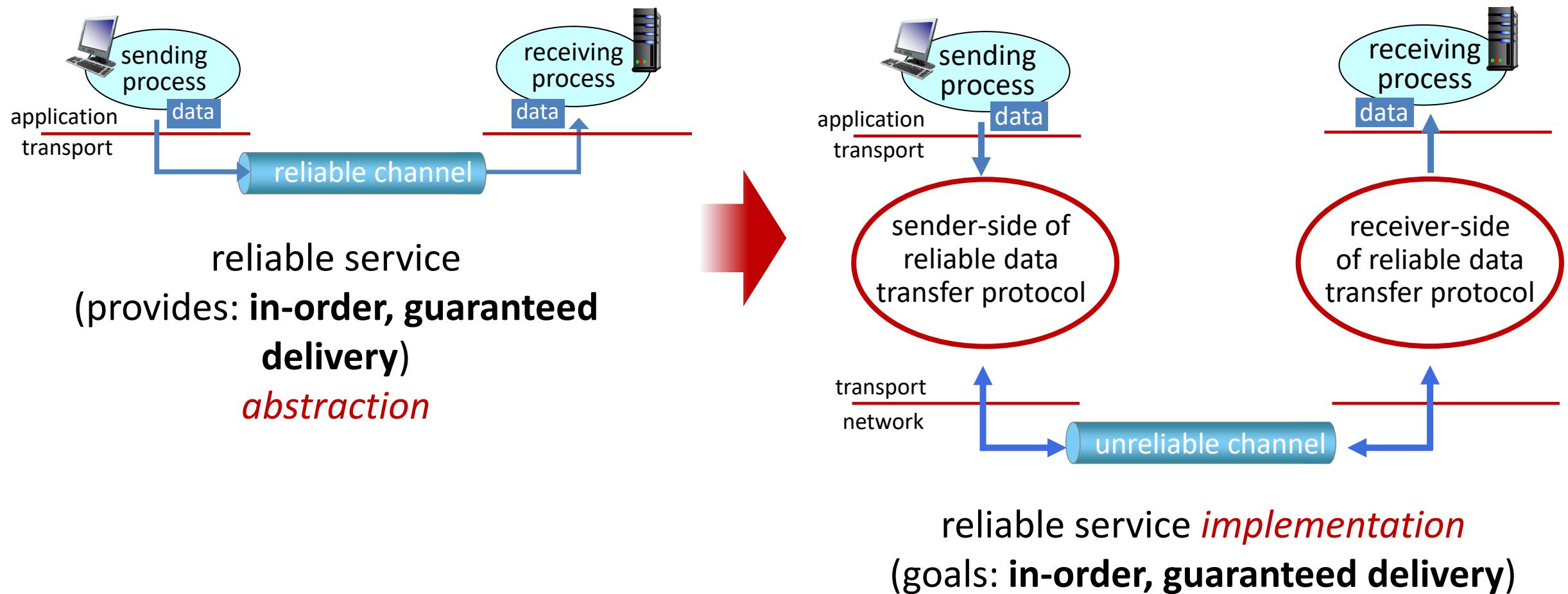
Roadmap



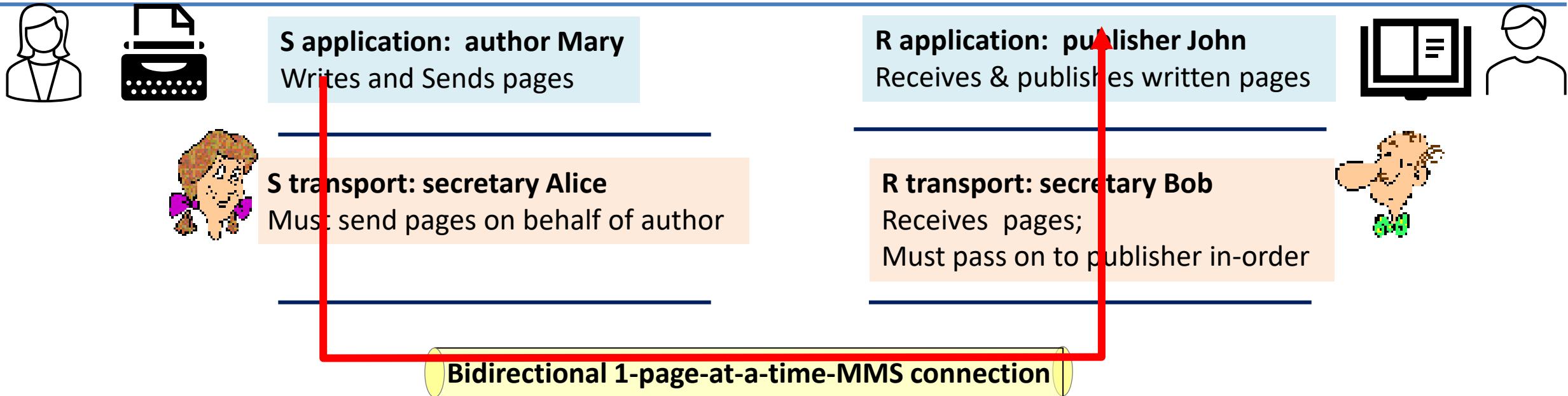
- Transport layer services
- Addressing, multiplexing/demultiplexing
- Principles of reliable data transfer
 - Outline: goal and common-practice approach
- Connectionless, unreliable transport: UDP

- *Next lecture: connection-oriented transport: TCP*
 - *reliable transfer in practice + more stuff TCP does*

Principles of reliable data transfer (RDT)



RDT (reliable data transfer) :



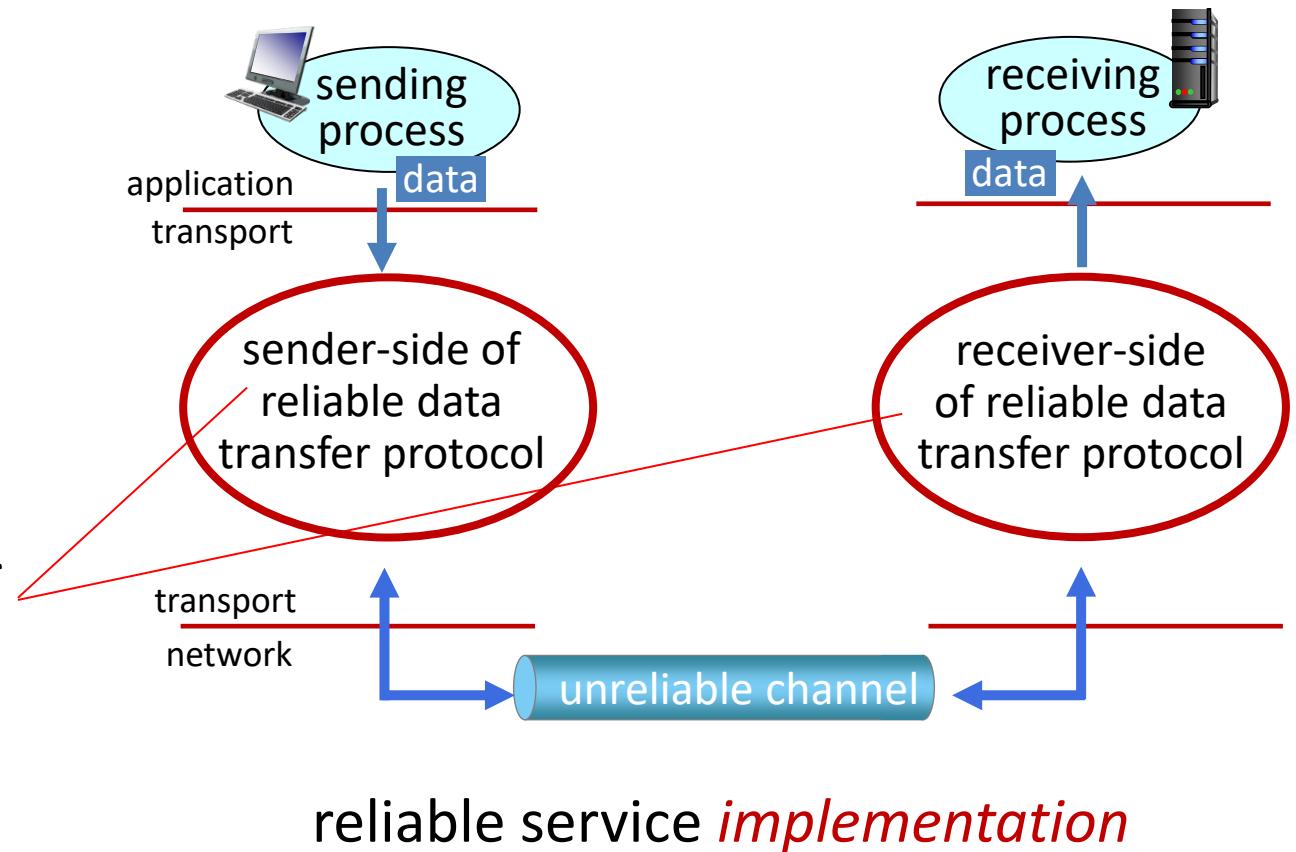
A sends one page at a time;

How do A & B do their job if MMS connection...

- ...is reliable?
- ...might introduce (detectable) errors?
- ...might lose MMSs?

Principles of reliable data transfer (RDT)

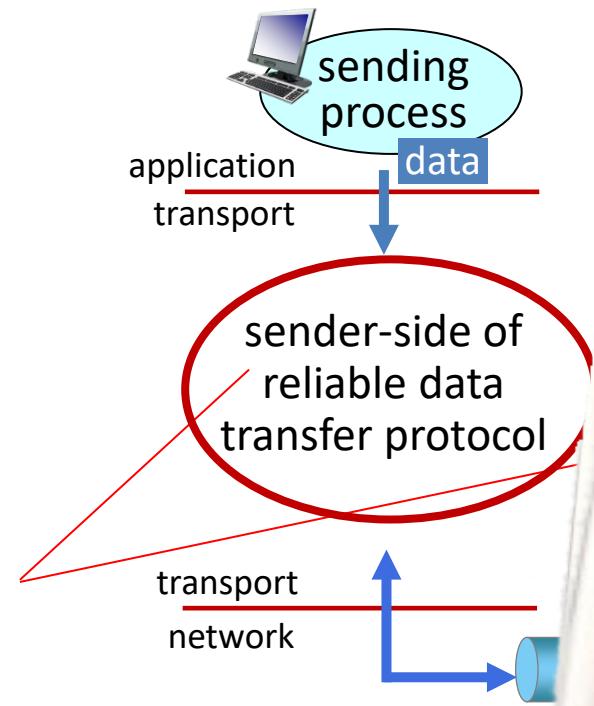
Complexity of reliable data transfer protocol will depend (strongly) on characteristics of unreliable channel (can it lose, corrupt, reorder data?)



Principles of reliable data transfer (RDT)

Sender, receiver do *not* know the “state” of each other, e.g., was a message received?

- unless communicated via a message



reliable service *implementation*

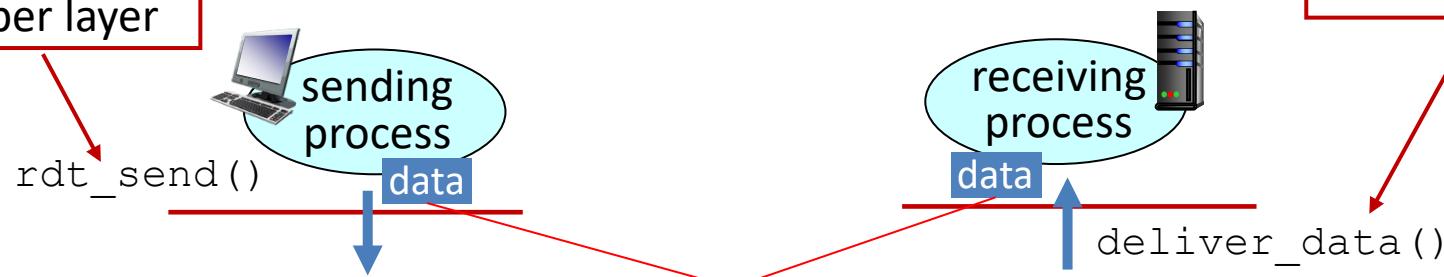
Roadmap



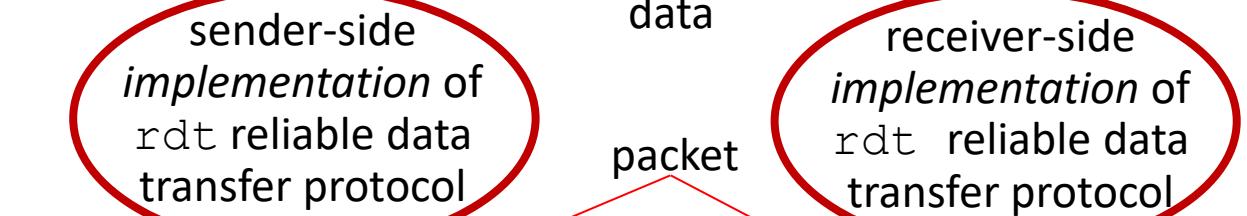
- Transport layer services
- Addressing, multiplexing/demultiplexing
- Principles of reliable data transfer (rdt)
 - Outline: goal and common-practice approach
 - Simple rdt with a bit of formalism
- Connectionless, unreliable transport: UDP
- *Next lecture: connection-oriented transport: TCP*
 - *reliable transfer in practice + more stuff TCP does*

Reliable data transfer protocol (rdt): interfaces

rdt_send(): called from above, (e.g., by app.). Passed data to deliver to receiver upper layer



deliver_data(): called by rdt to deliver data to upper layer



udt_send(): called by rdt to transfer packet over unreliable channel to receiver

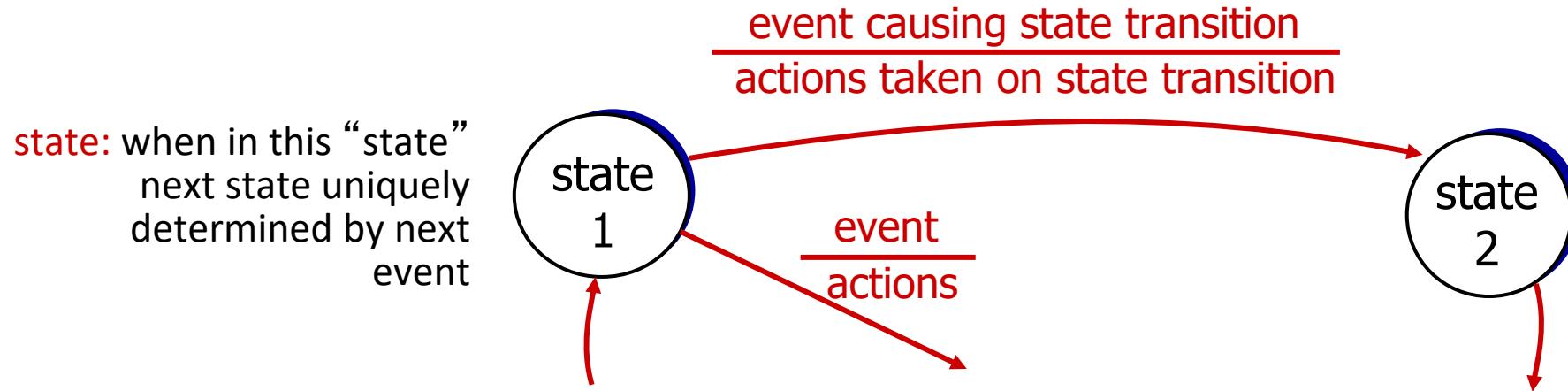
Bi-directional communication over unreliable channel

rdt_rcv(): called when packet arrives on receiver side of channel

Reliable data transfer: getting started

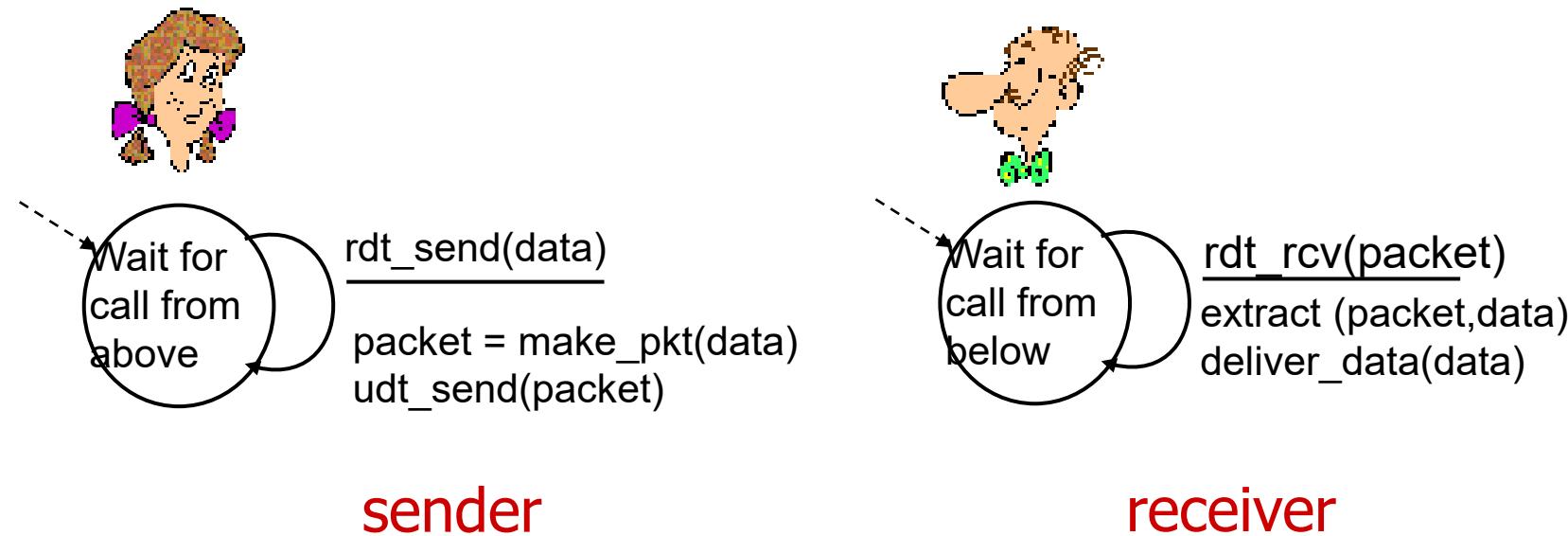
We will:

- incrementally develop sender, receiver sides of reliable data transfer protocol (rdt)
- consider only unidirectional data transfer
 - but control info will flow in both directions!
- use finite state machines (FSM) to specify sender, receiver



rdt1.0: reliable transfer & reliable channel

- underlying channel perfectly reliable
 - no bit errors, no loss of packets
- separate FSMs for sender, receiver:



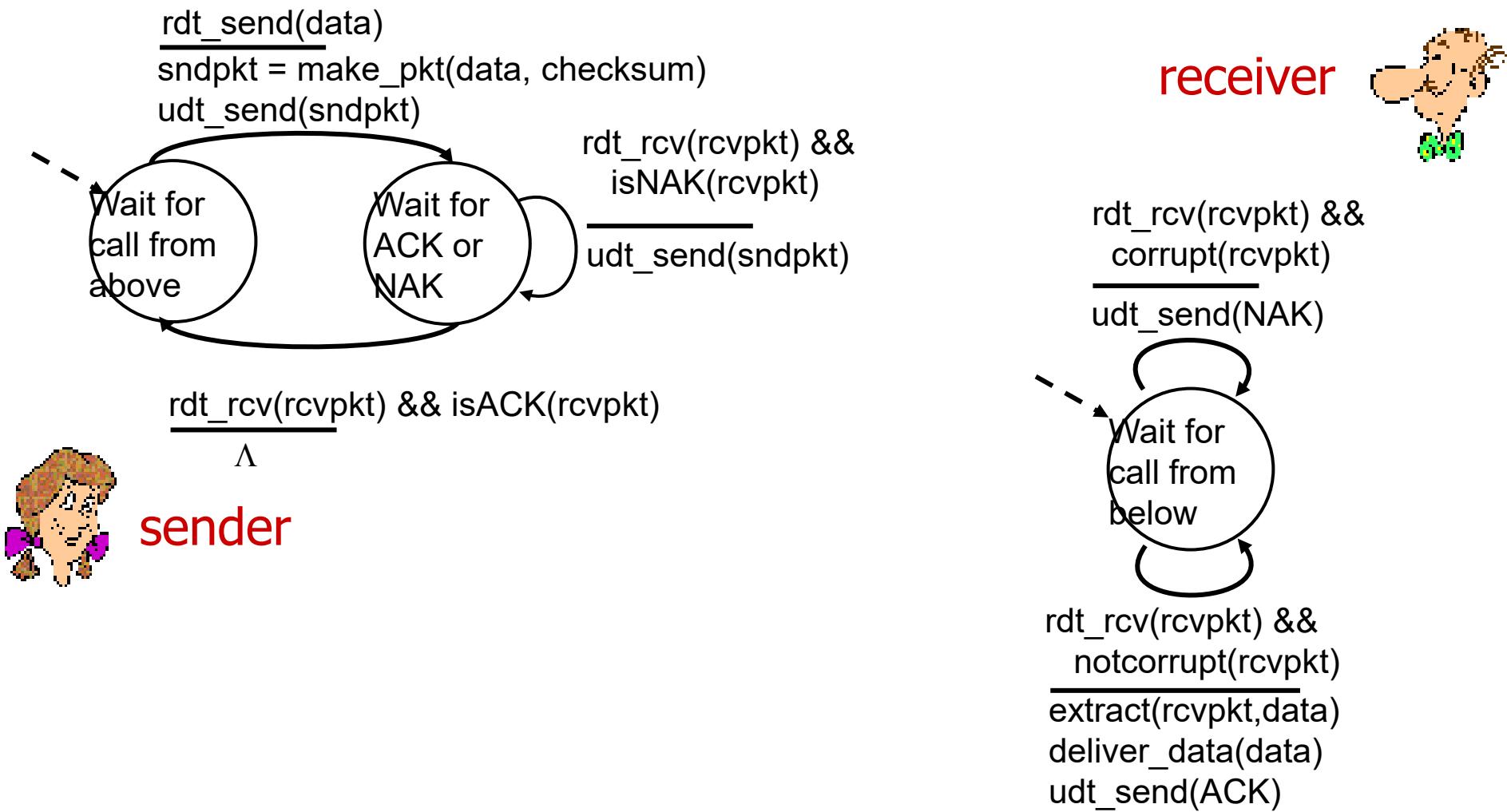
rdt2.0: channel with bit errors

- underlying channel may flip bits in packet
 - checksum to detect bit errors
- how to recover from errors:
 - *acknowledgements (ACKs)*: receiver explicitly tells sender that pkt received OK
 - *negative acknowledgements (NAKs)*: receiver explicitly tells sender that pkt had errors
 - sender retransmits pkt on receipt of NAK
- Note: This is one form of error control!

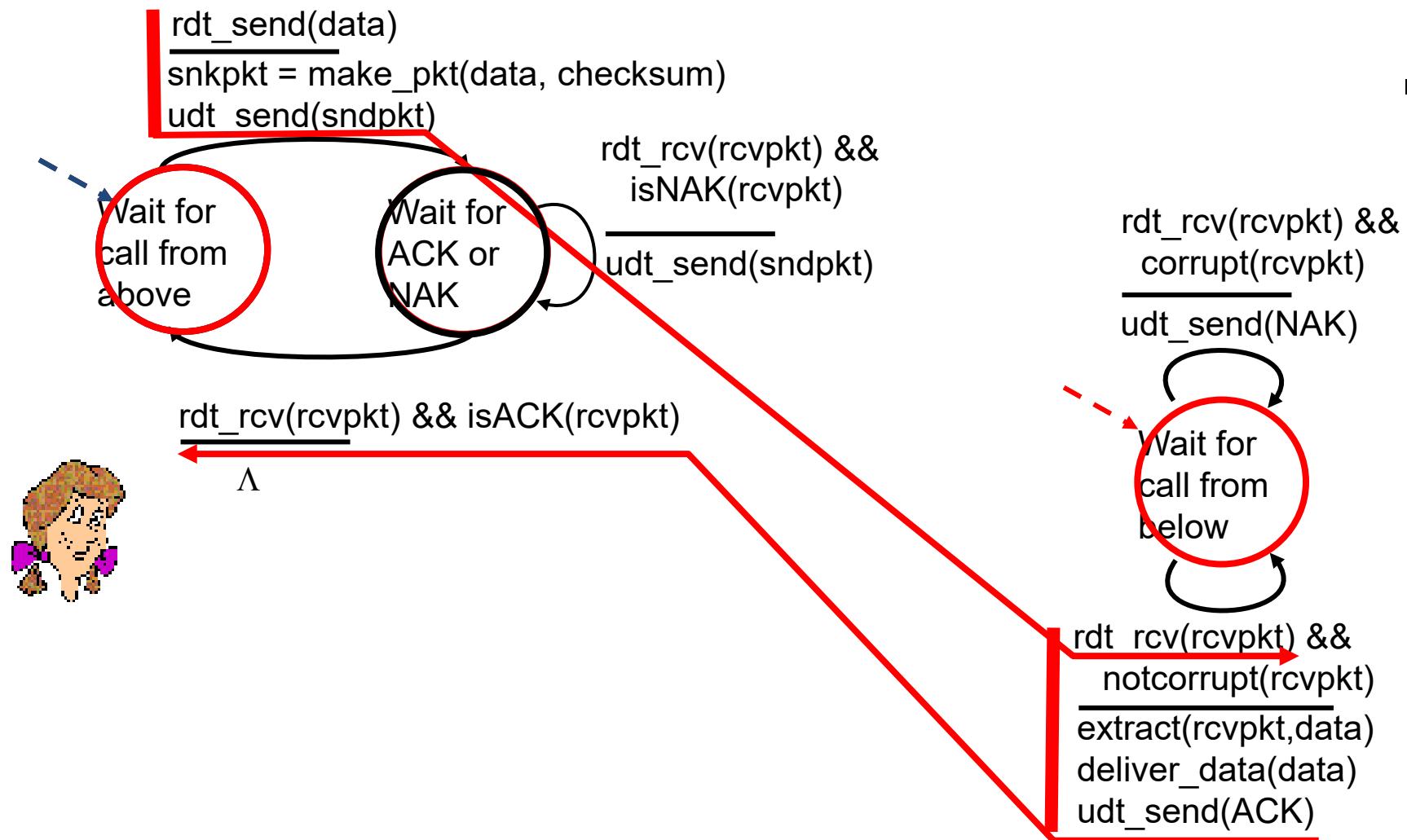
New mechanisms in **rdt2 . 0** (beyond **rdt1 . 0**):

- error detection
- feedback: control msgs (ACK,NAK) from receiver to sender

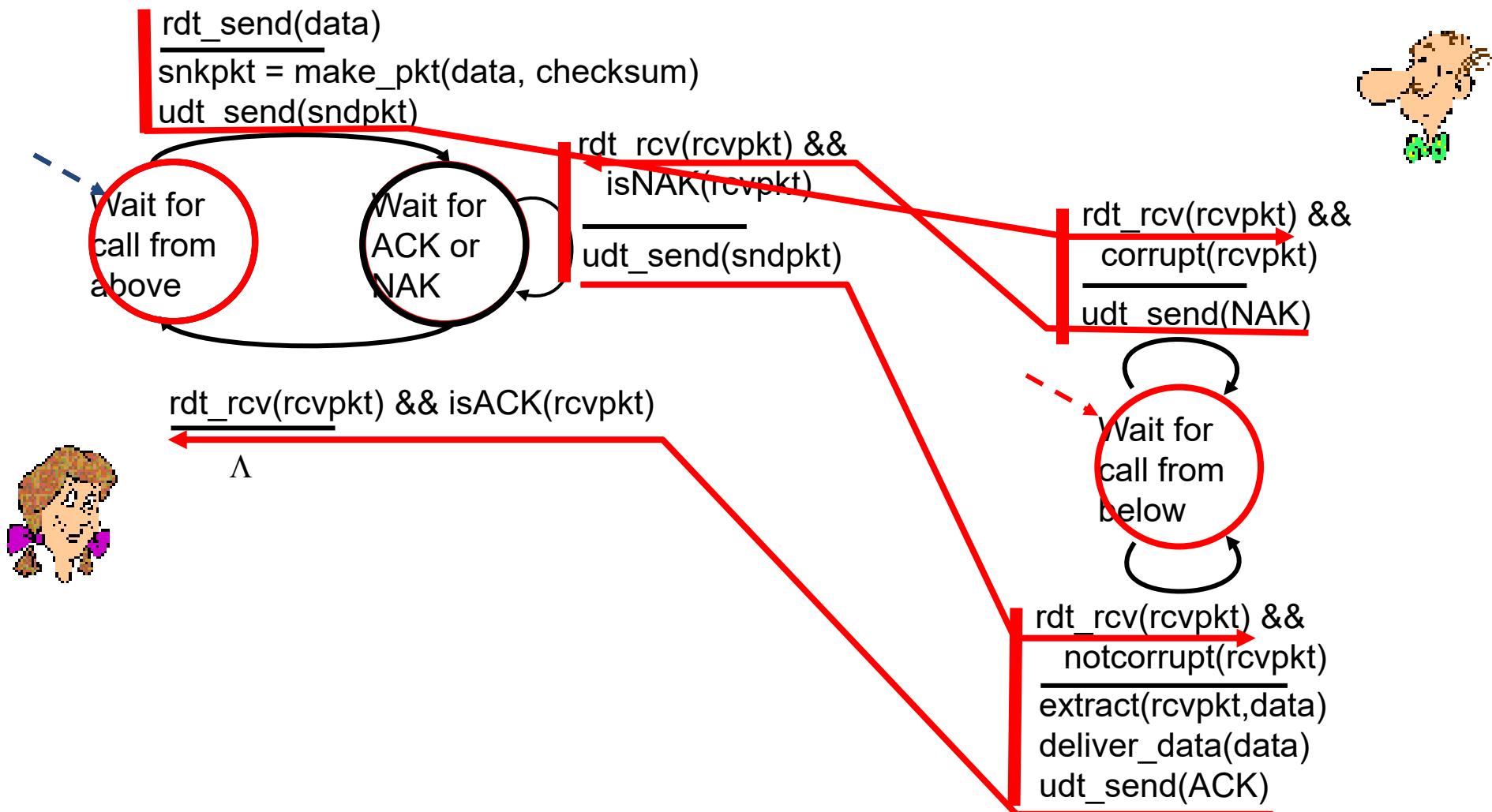
rdt2.0: FSM specification



rdt2.0: operation with no errors



rdt2.0: error scenario



Roadmap



- Transport layer services
 - Addressing, multiplexing/demultiplexing
 - Principles of reliable data transfer
 - Outline: goal and common-practice approach
 - Simple rtd with a bit of formalism
 - ... and with one more degree of difficulty
 - Connectionless, unreliable transport: UDP
-
- *Next lecture: connection-oriented transport: TCP*
 - *reliable transfer in practice + more stuff TCP does*

Recall RDT:



S application: Author

Writes and Sends pages



S transport: secretary Alice

Must send pages on behalf of author



R application: publisher

Receives & publishes written pages



S transport: secretary Bob

Receives pages;
Must pass on to publisher in-order

Bidirectional 1-page-at-a-time-MMS connection

A sends one page at a time;

How do A & B do their job if MMS connection...

- ...is reliable?
- ...might introduce errors?
- ...might lose MMSs?

rdt3.0: channels with errors *and* loss

We saw: how ack+retransmit can solve problems with errors

If underlying channel can also **lose MMS's** (data, ACKs)....

approach: sender waits “reasonable” amount of time for ACK

- retransmits if no ACK received in this time
 - requires countdown timer



Recall: RDT

(Reliable Data Transfer, aka error control)



S application: Author

Writes and Sends pages



S transport: secretary Alice

Must send pages on behalf of author



R application: publisher

Receives & publishes written pages



S transport: secretary Bob

Receives pages;
Must pass on to publisher in-order

Bidirectional 1-page-at-a-time-MMS connection

A sends one page at a time;

How do A & B do their job if MMS connection...

- ...is reliable?
- ...might introduce errors?
- ...might lose MMSs?
 - (lost vs too late MMS?)

rdt3.0 (cont) : channels with errors *and* loss

We saw: how ack+retransmit can solve problems with errors

If underlying channel can also **lose** packets (data, ACKs)

approach: sender waits “reasonable” amount of time for ACK

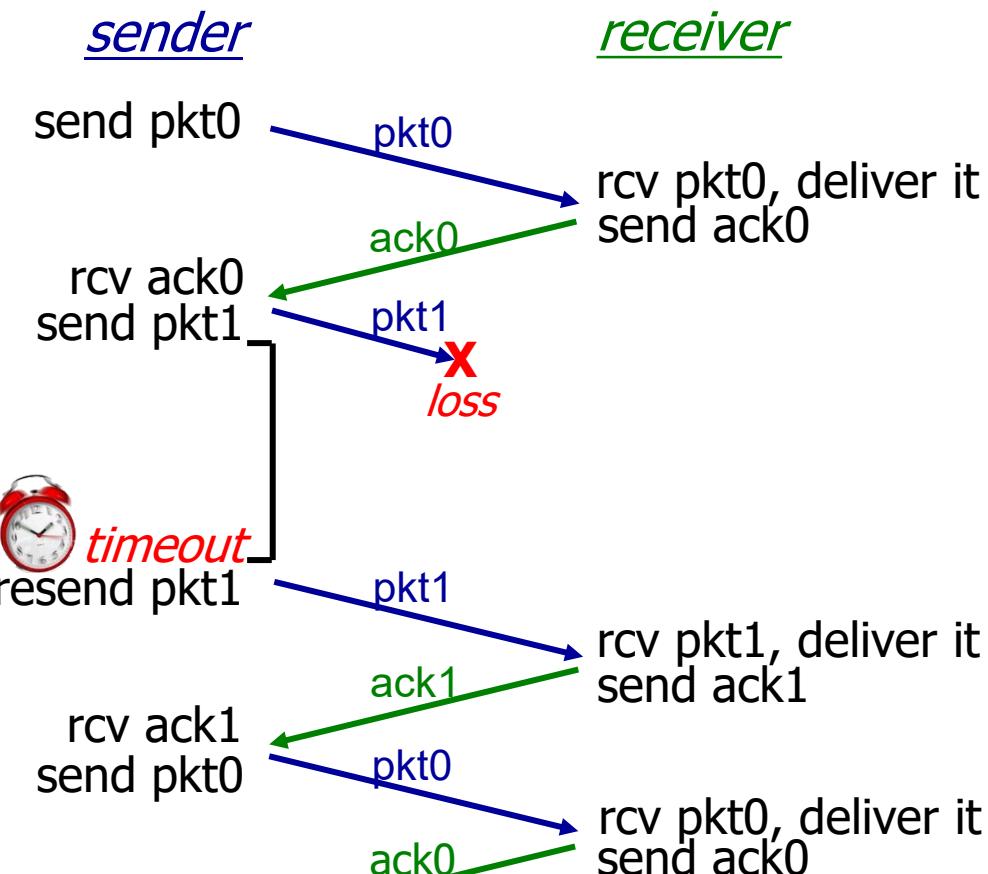
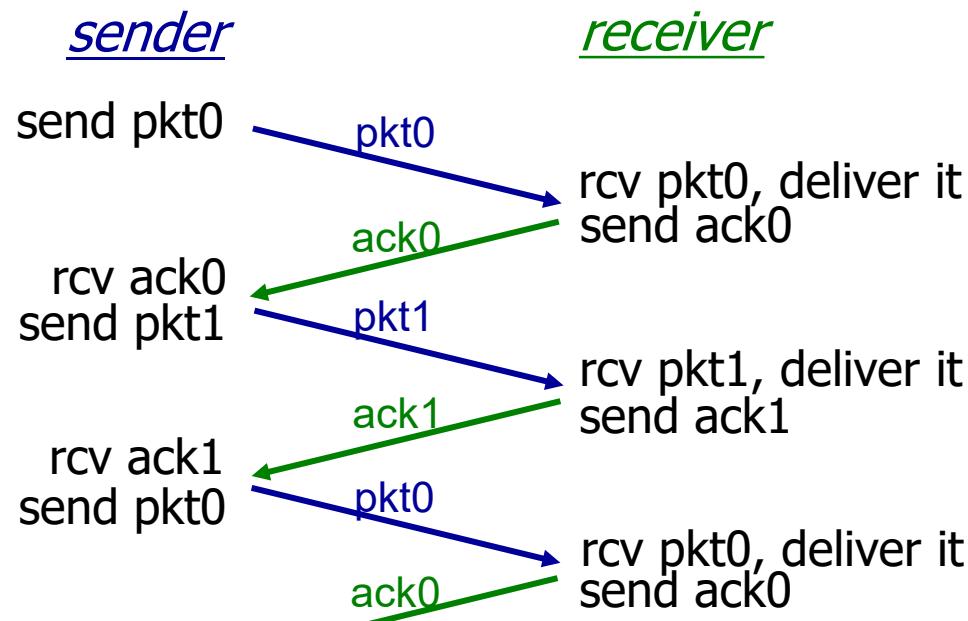
- retransmits if no ACK received in this t
 - requires countdown timer
- if pkt (or ACK) just delayed (not lost):
 - Must handle duplicates ->



handling duplicates:

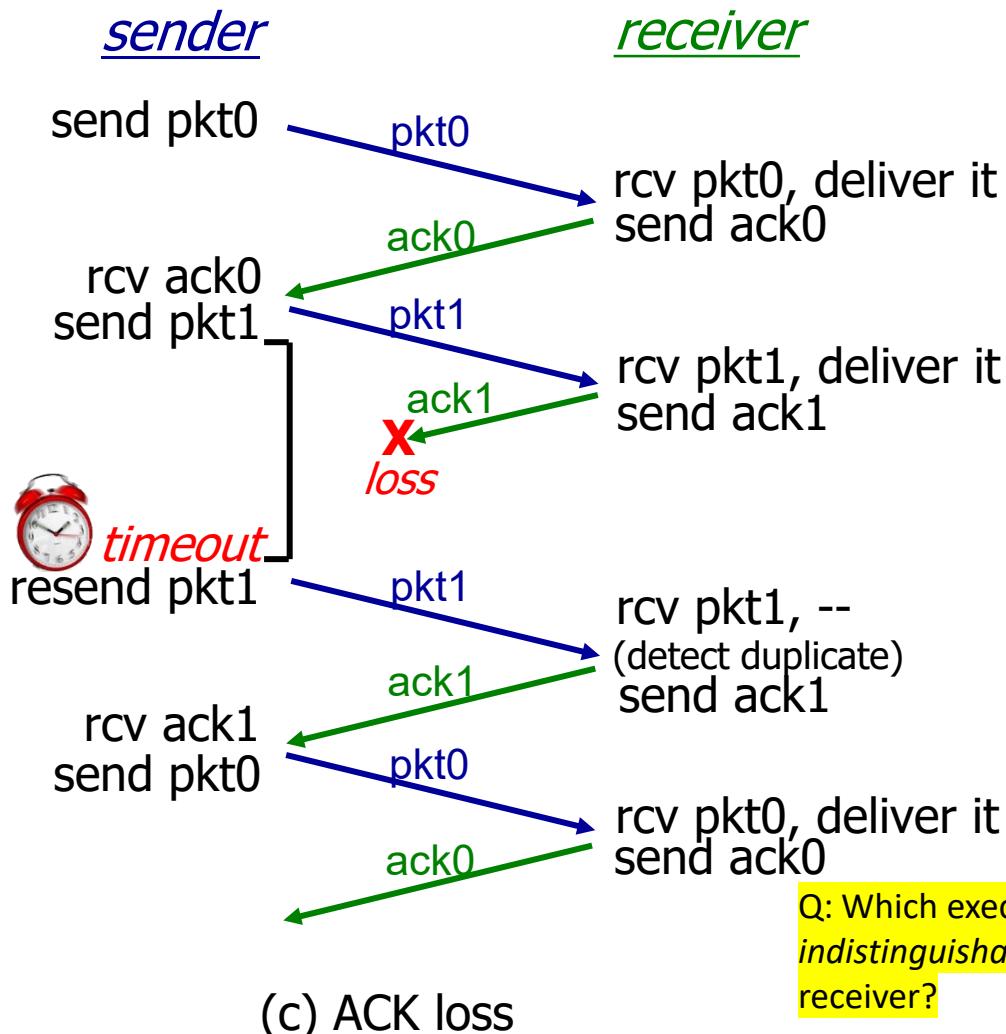
- sender adds **sequence number** to each pkt
- receiver **discards** duplicate pkt (doesn't deliver upwards)
- For our simple stop&wait protocol: 0-1 values (ie 1 bit) enough for sequence nr.

rdt3.0 in action

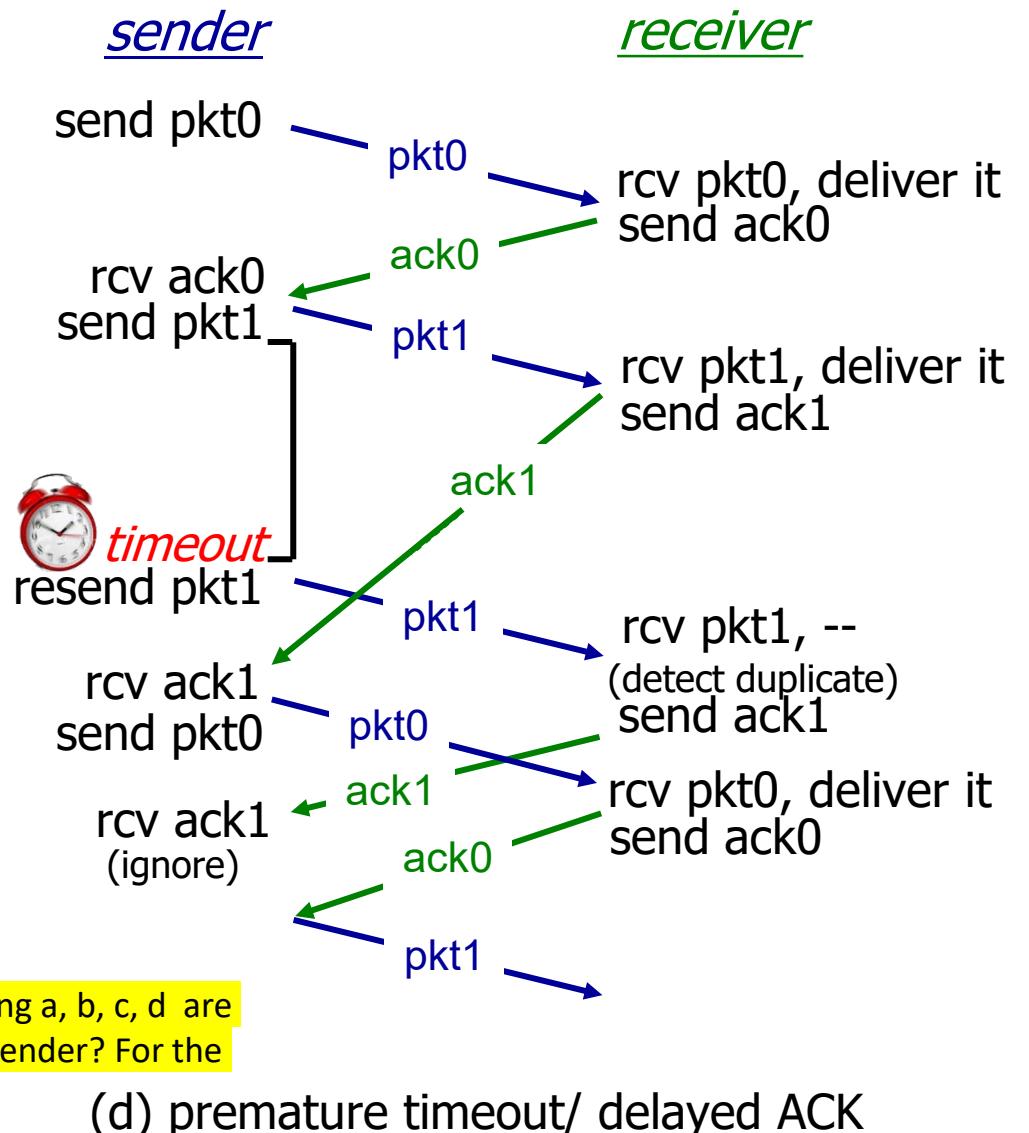


(b) packet loss

rdt3.0 in action (cont)



Q: Which executions among a, b, c, d are indistinguishable for the sender? For the receiver?



Roadmap

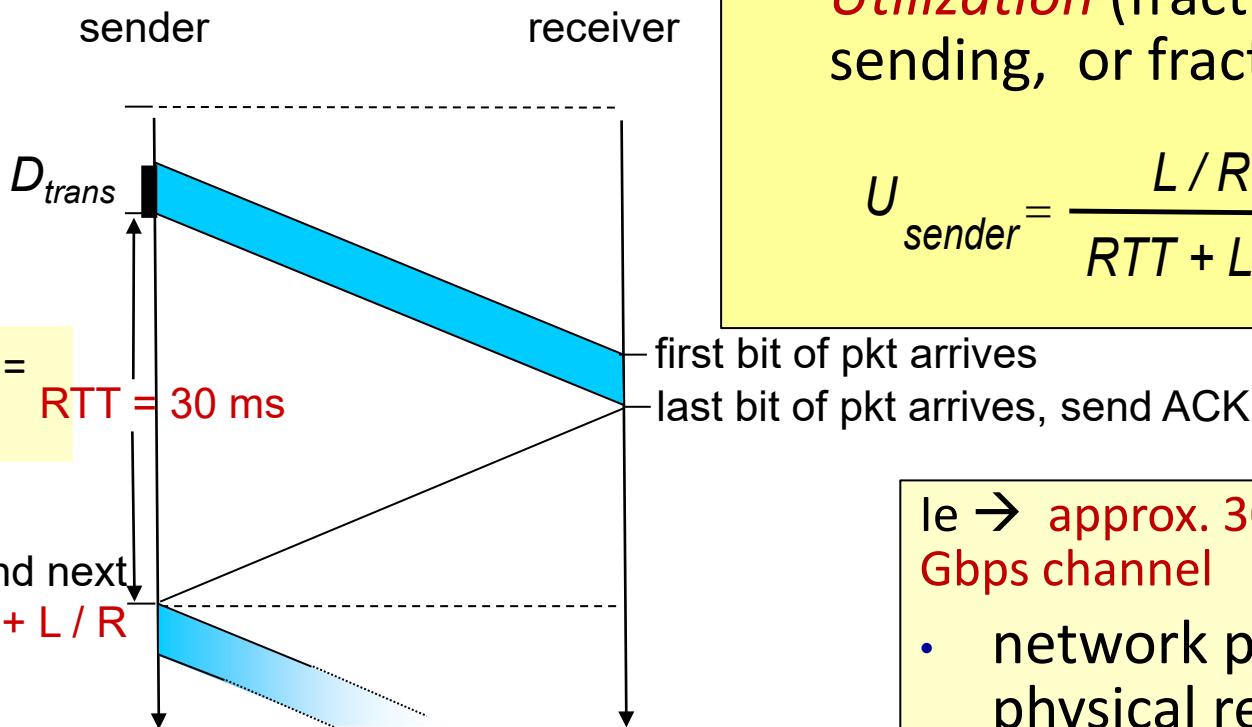


- Transport layer services
- Addressing, multiplexing/demultiplexing
- Principles of reliable data transfer
 - Outline: goal and common-practice approach
 - Simple rtd with a bit of formalism
 - ... and with one more degree of difficulty
 - Efficiency perspective
- Connectionless, unreliable transport: UDP
- *Next lecture: connection-oriented transport: TCP*
 - *reliable transfer in practice + more stuff TCP does*

Performance of rdt3.0 (stop&wait)

e.g.: R=1 Gbps channel, L= 8000 (1KB) bit packet length, $D_{prop} = 15 \text{ msec}$:

$$D_{trans} = \frac{L}{R} = \frac{8000 \text{ bits}}{10^9 \text{ bits/sec}} = 0.008 \text{ msec}$$



Utilization (fraction of time sender busy sending, or fraction of utilized bandwidth):

$$U_{\text{sender}} = \frac{L / R}{RTT + L / R} = \frac{.008}{30.008} = 0.00027$$

Ie → approx. 300 Kbps throughput over a 1 Gbps channel

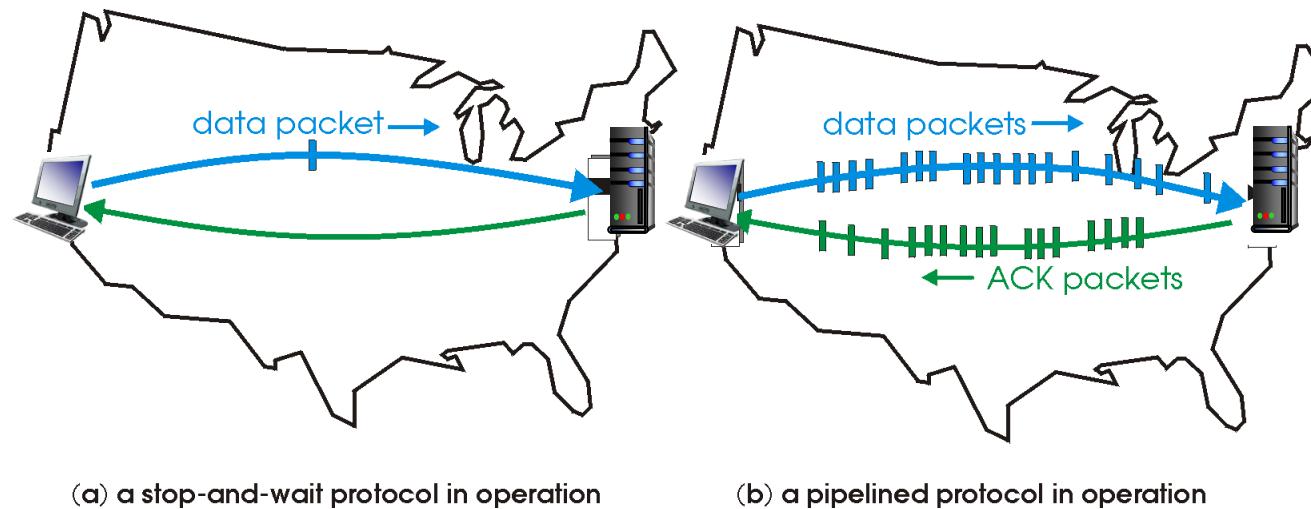
- network protocol limits use of physical resources!

Is RDT necessarily that slow/inefficient?

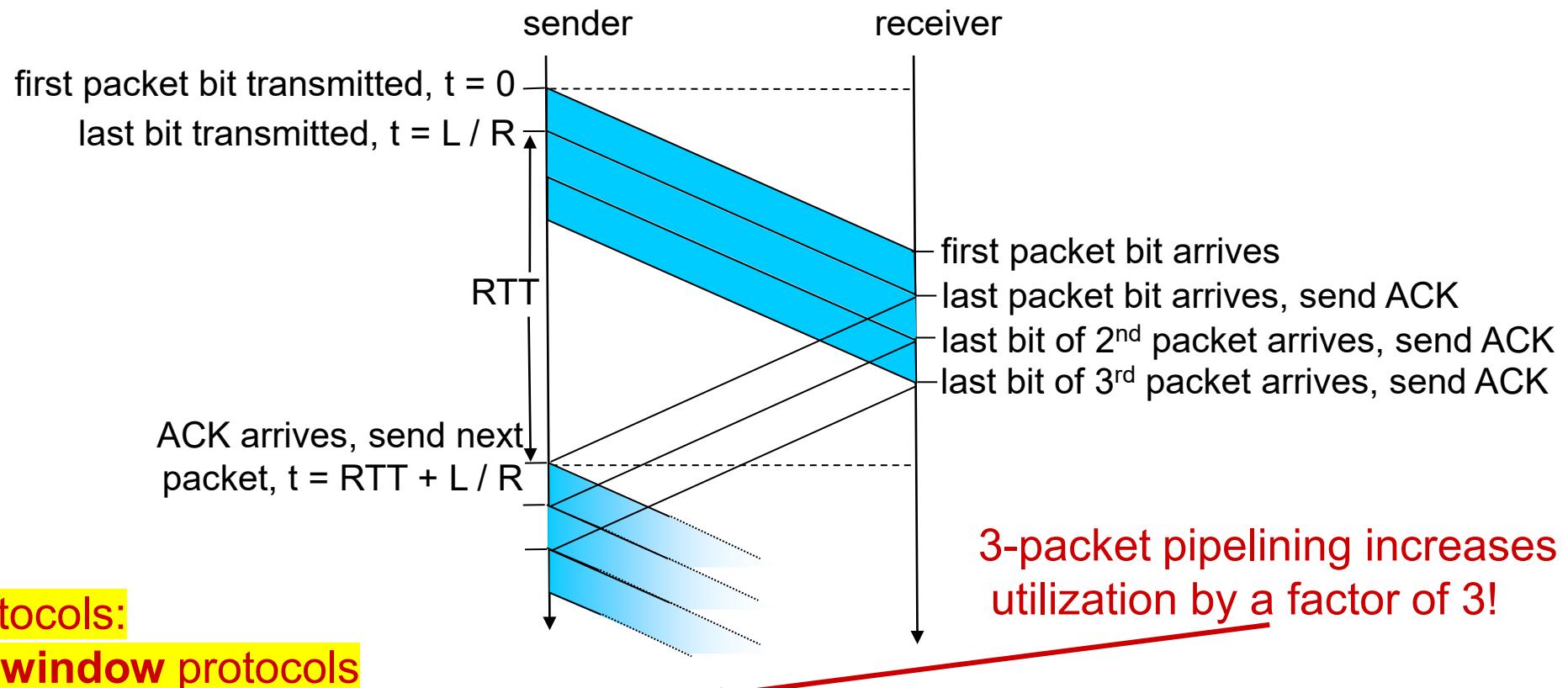
Pipelined protocols

pipelining: sender allows multiple, “in-flight”, yet-to-be-acknowledged pkts

- range of sequence numbers must be increased
- buffering at sender and/or receiver



Pipelining: increased utilization



$$U_{\text{sender}} = \frac{3L/R}{\text{RTT} + L/R} = \frac{.0024}{30.008} = 0.00081$$

Go to www.menti.com and use the code 6145 9030

Consider pkts of 10 Kbits, bandwidth 10 Mbps, RTT 10ms. What is the channel utilization if we allow arbitrarily high level of pipelining?



Roadmap



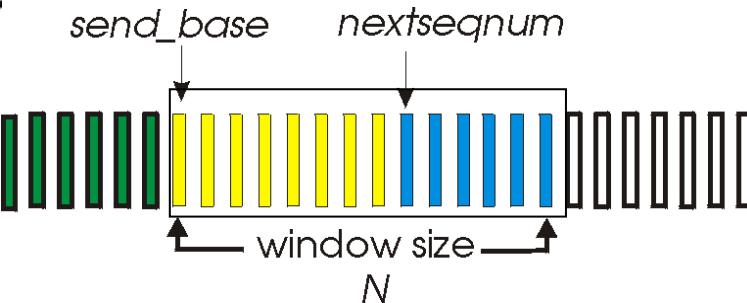
- Transport layer services
- Addressing, multiplexing/demultiplexing
- Principles of reliable data transfer
 - Outline: goal and common-practice approach
 - Simple rtd with a bit of formalism
 - ... and with one more degree of difficulty
 - Efficiency perspective
 - Error control in the (more efficient) pipelined protocols
- Connectionless, unreliable transport: UDP
- *Next lecture: connection-oriented transport: TCP*
 - *reliable transfer in practice + more stuff TCP does*

Error control in pipelined protocols

if data is lost, two generic forms of **ack-based error-control** in pipelined protocols: *go-Back-n, selective repeat*

Go-Back-n: sender

Sender view of sequence number space:
up to N (aka **window-size**) , consecutive unack' ed pkts allowed



already
ack'ed
sent, not
yet ack'ed
usable, not
yet sent
not usable

Sender

- timer for oldest in-flight pkt
- *timeout(x)*: **retransmit packet x and all higher seq # pkts in window**

Receiver

- ACK for correctly-received pkt with highest in-order seq #
- on receipt of out-of-order pkt:
 - re-ACK pkt with highest in-order seq #



Receiver view of sequence number space:



GBn in action

https://media.pearsoncmg.com/ph/esm/ecs_kurose_compnetwork_8/cw/content/interactiveanimations/go-back-n-protocol/index.html

sender window ($N=4$)

0	1	2	3	4	5	6	7	8
0	1	2	3	4	5	6	7	8
0	1	2	3	4	5	6	7	8
0	1	2	3	4	5	6	7	8

0	1	2	3	4	5	6	7	8
0	1	2	3	4	5	6	7	8

0	1	2	3	4	5	6	7	8
0	1	2	3	4	5	6	7	8
0	1	2	3	4	5	6	7	8
0	1	2	3	4	5	6	7	8

sender

send pkt0
send pkt1
send pkt2
send pkt3
(wait)



pkt 2 timeout

send pkt2
send pkt3
send pkt4
send pkt5

receiver

receive pkt0, send ack0
receive pkt1, send ack1

receive pkt3, discard,
(re)send ack1

receive pkt4, discard,
(re)send ack1

receive pkt5, discard,
(re)send ack1

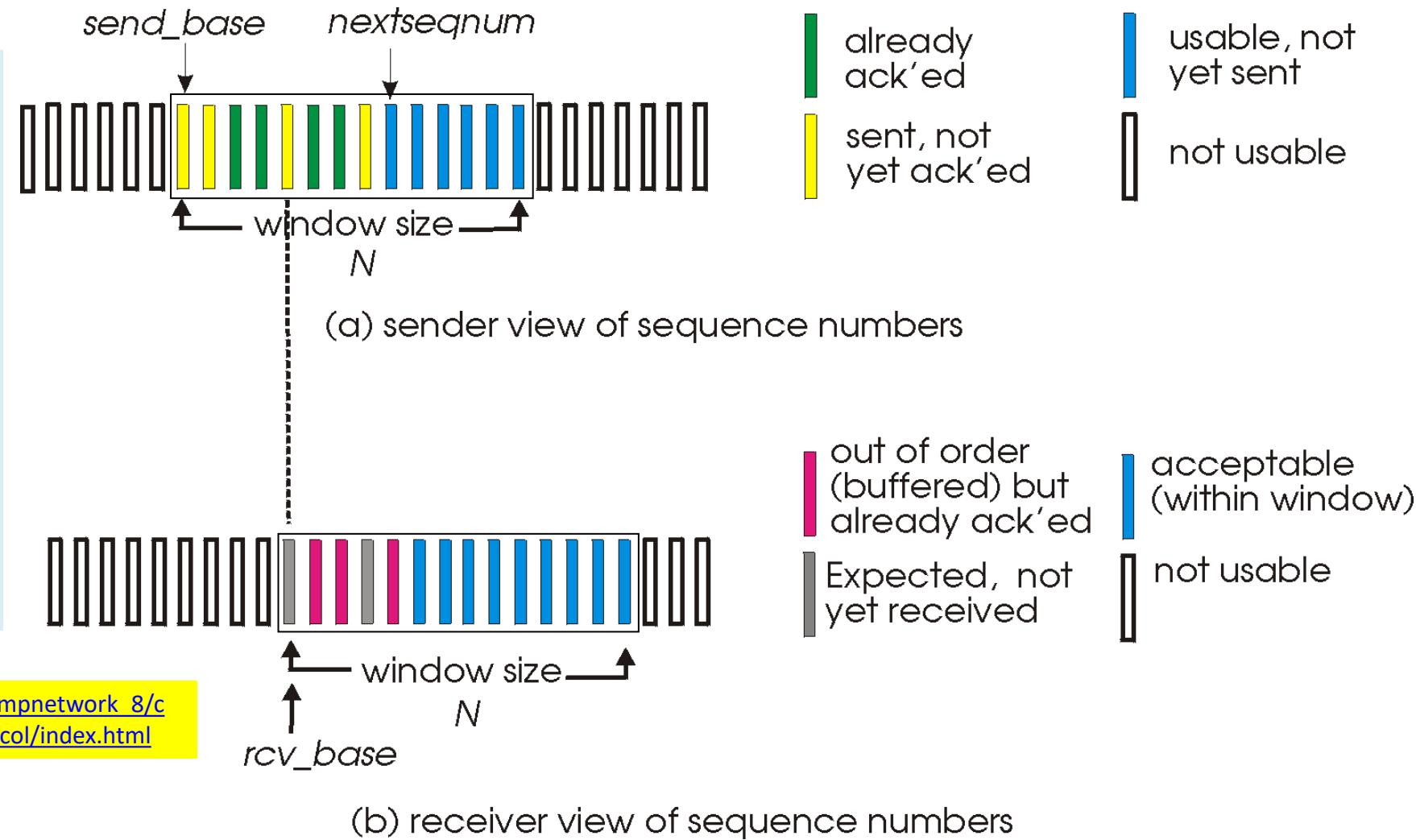
rcv pkt2, deliver, send ack2
rcv pkt3, deliver, send ack3
rcv pkt4, deliver, send ack4
rcv pkt5, deliver, send ack5

Selective repeat: sender, receiver windows

- sender only resends pkts for which ACK not received
 - Requires **timer for each unACKed pkt**
- receiver *individually* acknowledges received pkts
 - buffers pkts for eventual in-order delivery to upper layer



https://media.pearsoncmg.com/ph/esm/ecs_kurose_compnetwork_8/cw/content/interactiveanimations/selective-repeat-protocol/index.html



Selective repeat



Sender: upon...

- ...data from above:
 - if next_pkt_seq # in window, send pkt
- ...timeout(n):
 - resend pkt n, restart timer
- ...ACK(n) in [sendbase,sendbase+N]:
 - mark pkt n as received
 - if n smallest unACKed pkt, advance window base to next unACKed seq #

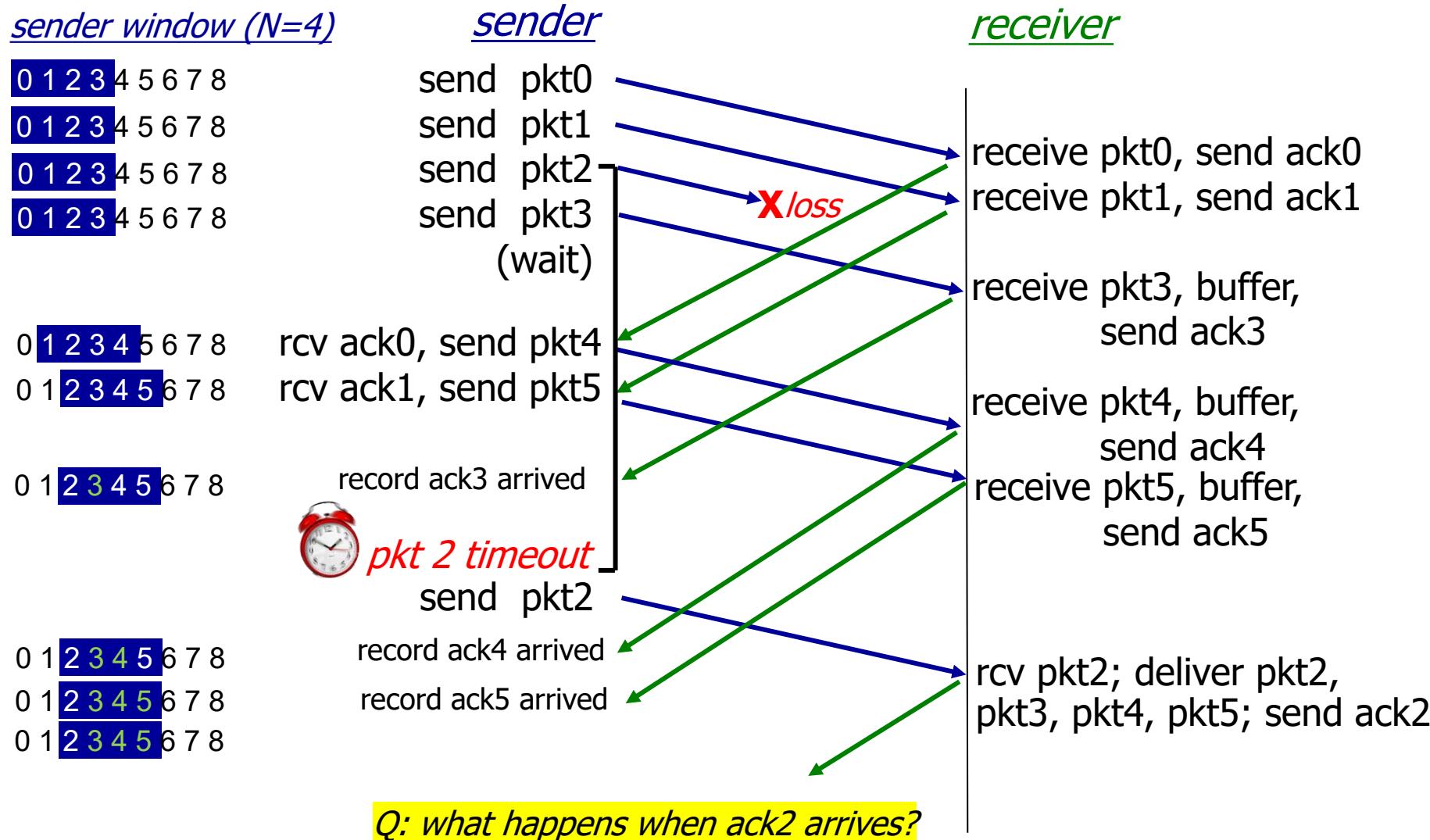


Receiver: upon receiving...

- ... pkt n in [rcvbase, rcvbase+N-1]
 - send ACK(n)
 - If out-of-order: buffer
 - If in-order: deliver (also deliver buffered, in-order pkts), advance window to next not-yet-received pkt
- ...pkt n in [rcvbase-N,rcvbase-1] // already ack-ed
 - ACK(n) // duplicate ack
- otherwise: // “future” window packets
 - ignore

Selective repeat in action

https://media.pearsoncmg.com/aw/ecs_kurose_compnetwork_7/cw/content/interactiveanimations/selective-repeat-protocol/index.html



Roadmap



- Transport layer services in Internet
- Addressing, multiplexing/demultiplexing
- Principles of reliable data transfer
 - Outline: goal and common-practice approach
 - Simple rtd with a bit of formalism
 - ... and with one more degree of difficulty
 - Efficiency perspective: pipelined protocols & error control through go-back-n, selective-repeat
 - Sequence numbers
- Connectionless, unreliable transport: UDP
- *Next: connection-oriented transport: TCP*
 - *reliable transfer + other good stuff that TCP does*

Sequence numbers

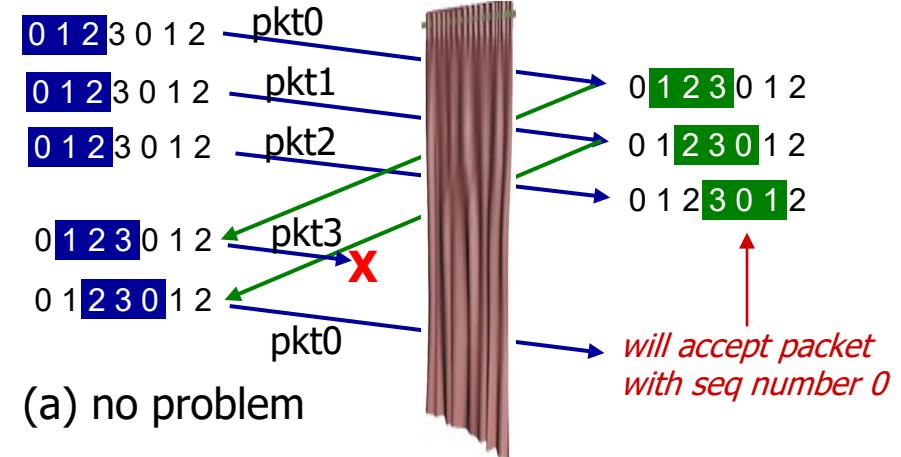
example:

- seq #'s: 0, 1, 2, 3
- window size=3
 - ❖ duplicate data accepted as new in (b)

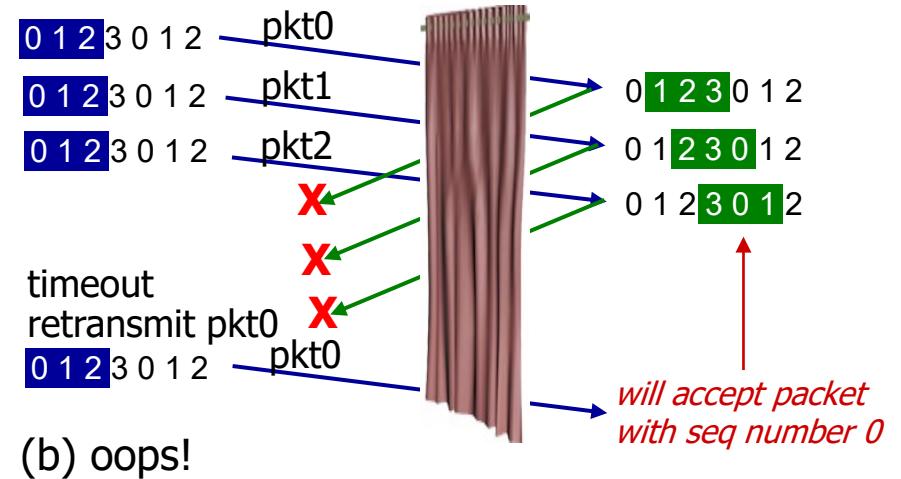
Q: relation between seq # size and window size to avoid problem in (b)?

sender window
(after receipt)

receiver window
(after receipt)



*receiver can't see sender side.
receiver behavior identical in both cases!
something's (very) wrong!*



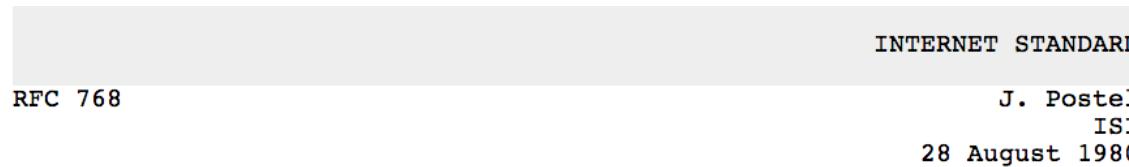
Roadmap



- Transport layer services
- Addressing, multiplexing/demultiplexing
- principles of reliable data transfer
- Connectionless, unreliable transport: UDP

- *Next lecture: connection-oriented transport: TCP*
 - *reliable transfer in practice + more stuff TCP does*

UDP: User Datagram Protocol [RFC 768]



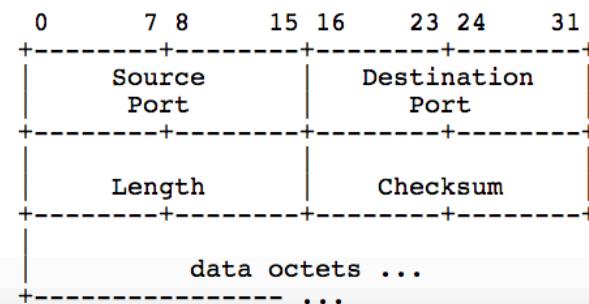
User Datagram Protocol

Introduction

This User Datagram Protocol (UDP) is defined to make available a datagram mode of packet-switched computer communication in the environment of an interconnected set of computer networks. This protocol assumes that the Internet Protocol (IP) [1] is used as the underlying protocol.

This protocol provides a procedure for application programs to send messages to other programs with a minimum of protocol mechanism. The protocol is transaction oriented, and delivery and duplicate protection are not guaranteed. Applications requiring ordered reliable delivery of streams of data should use the Transmission Control Protocol (TCP) [2].

Format



UDP: User Datagram Protocol [RFC 768]

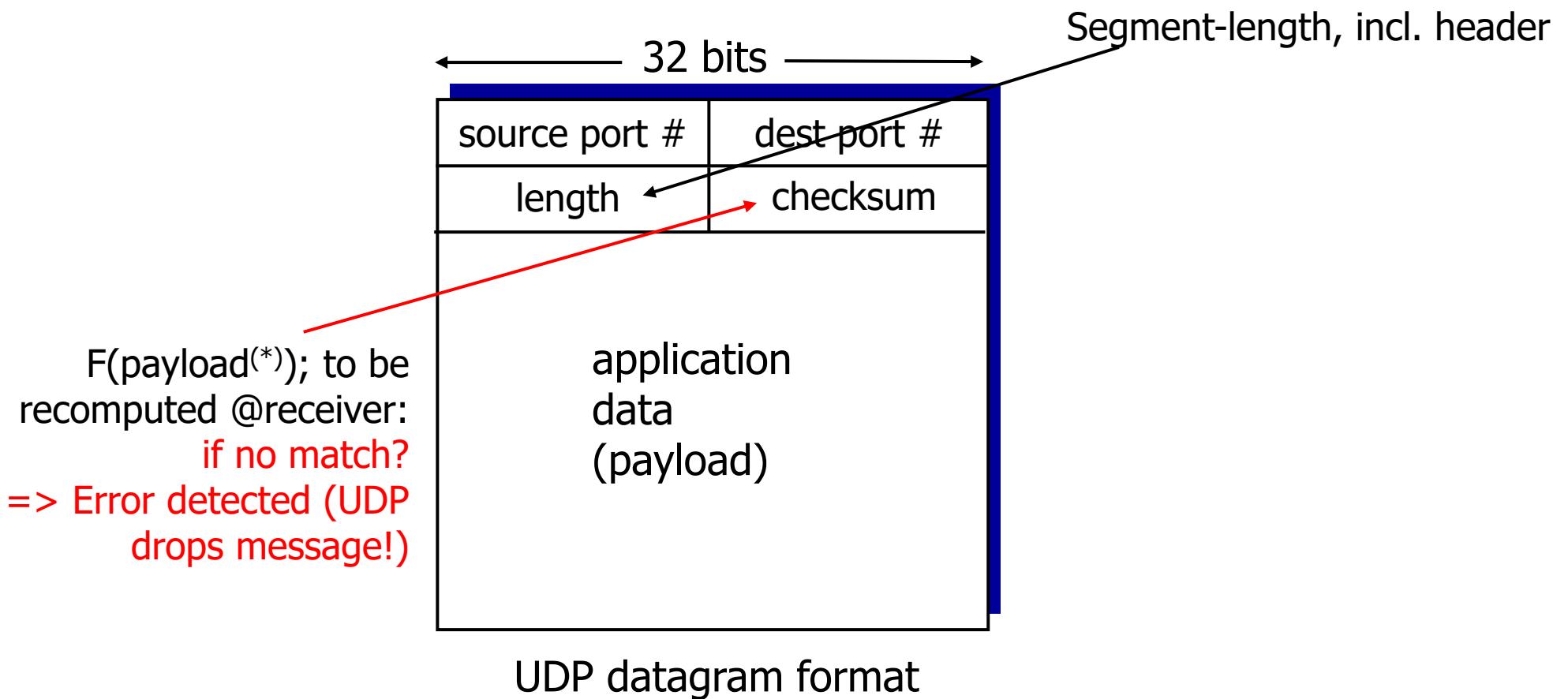
“best effort” service, UDP segments may be:

- lost
- delivered out-of-order
- *connectionless*:
 - no “handshaking” between UDP sender, receiver
 - each UDP segment handled independently of others

UDP used by:

- DNS
- SNMP
- More discussion later...

UDP: segment header



(*) addition (1's complement sum) of contents (payload), seen as seq. of 16-bit numbers

UDP Checksum [RFC 1071]

Sender:

- treat segment contents as sequence of 16-bit integers
- checksum: addition (1's complement sum) of segment contents
- sender puts checksum value into UDP checksum field

Receiver:

- compute checksum of received segment
- check if **computed checksum == checksum field** value:
 - NO - error detected (*report error to app or discard*)
 - YES - no error detected.
 - *But maybe (rarely) errors nonetheless?* More later

	1 1 1 0 0 1 1 0 0 1 1 0 0 1 1 0
	1 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1
<hr/>	
Wraparound: Add to final	<p>The diagram illustrates the 16-bit checksum calculation process. It shows two 16-bit binary numbers being added together. The result of the addition is then wrapped around to form a new 16-bit checksum. A red circle highlights the first bit of the result, and a red arrow points from the end of the result back to the beginning, indicating the wrap-around operation.</p> <p>1 1 0 1 1 1 0 1 1 1 0 1 1 1 0 1 1</p>
sum	1 0 1 1 1 0 1 1 1 0 1 1 1 1 0 0
checksum	0 1 0 0 0 1 0 0 0 1 0 0 0 0 1 1

Go to www.menti.com and use the code 6145 9030

Why have a UDP? (consider what services it offers)



Roadmap

- Transport layer services
 - Addressing, multiplexing/demultiplexing
 - Principles of reliable data transfer
 - Connectionless, unreliable transport: UDP
-
- *Next lecture: connection-oriented transport: TCP*
 - *reliable transfer*
 - *flow control*
 - *connection management*
 - *TCP congestion control*



Reading instructions chapter 3

- **KuroseRoss book**

Careful	Quick
3.1-3.7	

- **Other resources (further, optional study)**

- Lakshman, T. V., Upamanyu Madhow, and Bernhard Suter. "Window-based error recovery and flow control with a slow acknowledgement channel: a study of TCP/IP performance." INFOCOM'97. Sixteenth Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings IEEE. Vol. 3. IEEE, 1997.
- Rizzo, Luigi. "Effective erasure codes for reliable computer communication protocols." ACM SIGCOMM Computer Communication Review 27.2 (1997): 24-36.
- A. Agarwal and M. Charikar, “On the advantage of network coding for improving network throughput,” in Proceedings of the IEEE Information Theory Workshop, Oct. 2004
- Harvey, N. J., Kleinberg, R., & Lehman, A. R. (2006). On the capacity of information networks. IEEE/ACM Transactions on Networking (TON), 14(SI), 2345-2364.

Review question

MULTIPLEXING/DEMULITPLEXING & PORT NUMBERS.

- True or False: When *multiple UDP clients* send UDP segments to the *same destination port number at a receiving host*, those segments (from different senders) will always be directed to the *same socket at the receiving host*.
- True or False? When *multiple TCP clients* send TCP segments to the *same destination port number at a receiving host*, those segments (from different senders) will always be directed to the *same socket at the receiving host*.

Review question

Pipelining & Channel Utilization: Suppose a packet is 10K bits long, the channel transmission rate connecting a sender and receiver is 10 Mbps, and the round-trip propagation delay is 10 ms. What is the channel utilization of a pipelined protocol with an arbitrarily high level of pipelining for this channel?

1. 10.0
2. 0.01
3. 0.1
- 4.** 1.0
5. 0.001

Review question

GO-BACK-N. What are some reasons for discarding received-but-out-of-sequence packets at the receiver in GBN? Indicate one or more of the following statements that are correct.

1. The implementation at the receiver is simpler.
2. Discarding an out of sequence packet will really force the sender to retransmit.
3. If some packets are in error, then it's likely that other packets are in error as well.
4. The sender will resend that packet in any case.

Q: Why is there a UDP? (What if no UDP?)

(Role of UDP)

- must do the addressing (i.e. the transport admin) job
- does a very simple error-check
- simple: no connection state at sender, receiver, no other functionality, small header size, hence faster -- can blast away segments faster (than TCP)

Some more review questions on this part

- Why do we need an extra protocol, i.e. UDP, to deliver the datagram service of Internets IP to the applications?
- Draw space-time diagrams without errors and with errors, for the following, for a pair of sender-receive S-R: (assume only 1 link between them)
 - Stop-and-wait: transmission delay < propagation delay and transmission delay > propagation delay
 - Sliding window aka pipelined protocol, with window's transmission delay < propagation delay and window's transmission delay > propagation delay; illustrate both go-back-n and selective repeat when there are errors
 - Show how to compute the effective throughput between S-R in the above cases, when there are no errors
- What are the goals of reliable data transfer?
- Reliable data transfer: show why we need sequence numbers when the sender may retransmit due to timeouts.
- Show how there can be wraparound in a reliable data transfer session if the sequence-numbers range is not large enough.
- Describe the go-back-N and selective repeat methods for reliable data transfer

Bounding sequence numbers for stop-and-wait...

... s.t. **no wraparound**, i.e. we do not run out of numbers: *0 and 1 suffices for stop-and-wait:*

Key argument: assume towards a contradiction that there is wraparound when we use binary seq. nums.

- R expects segment # f , receives segment #($f+2$):
R rec. $f+2 \Rightarrow S$ sent $f+2 \Rightarrow S$ rec. ack for $f+1$
 $\Rightarrow R$ ack $f+1 \Rightarrow R$ ack $f \Rightarrow$ contradiction (*i.e. R cannot expect f*)
- R expects $f+2$, receives f :
R exp. $f+2 \Rightarrow R$ ack $f+1 \Rightarrow S$ sent $f+1$
 $\Rightarrow S$ rec. ack for $f \Rightarrow$ contradiction (*i.e. S cannot re-send f*)



Course on Computer Communication and Networks

Lecture 5 Chapter 3; Transport Layer, Part B: TCP

EDA344/DIT423/LEU062

Lecturer: Hans-Martin Heyn

Based on the book Computer Networking: A Top Down Approach, Jim Kurose, Keith Ross, Addison-Wesley.

Layering: The OSI Reference Model

ISO (International Standards Organization) defined the OSI (Open Systems Interconnect) model to help vendors create interoperable network implementation

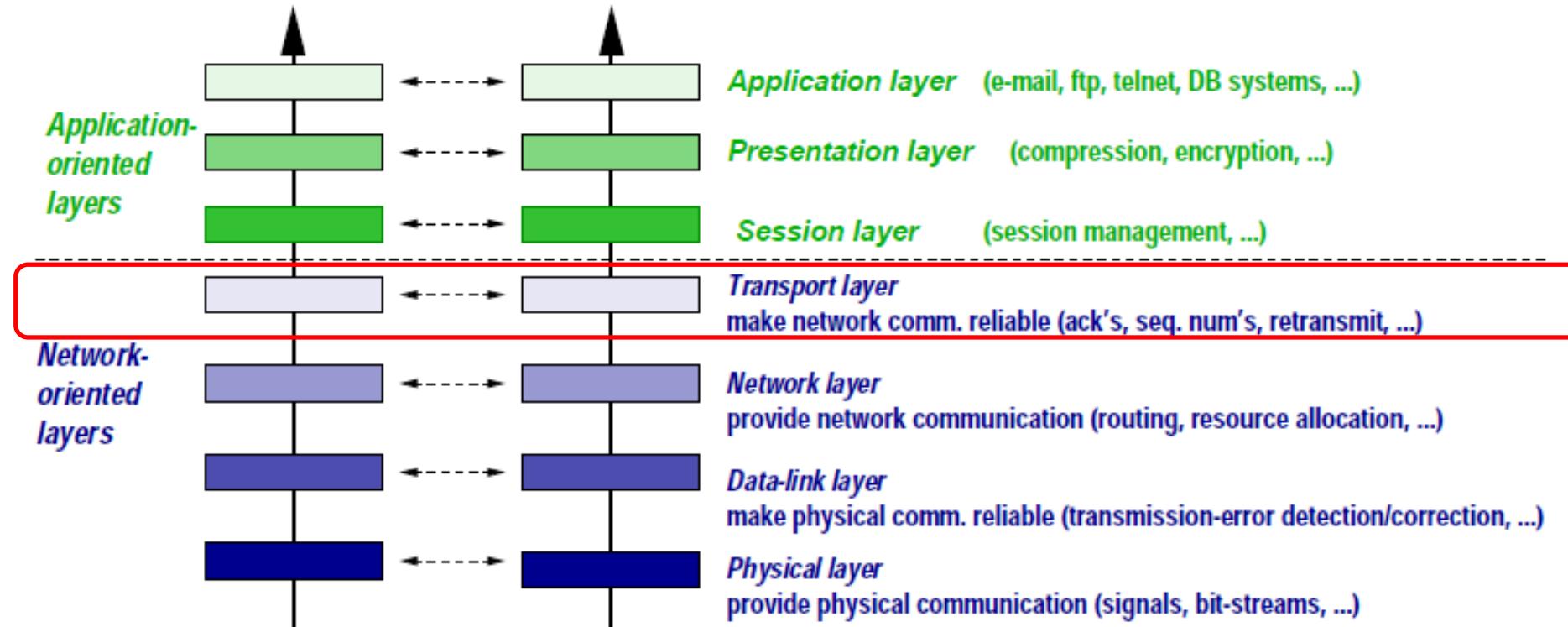


Fig. Steen, Sips : Computer and Network organization

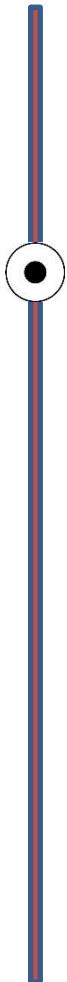
“X dot” series (X.25, X.400, X.500) OSI model implementation (protocol stack)

Roadmap Transport Layer

- transport layer services
- multiplexing/demultiplexing
- connectionless transport: UDP
- principles of reliable data transfer
- connection-oriented transport: TCP
 - Reliable transfer
 - Acknowledgements
 - Retransmissions
 - Connection management
 - Flow control and buffer space
 - Congestion control
 - Principles
 - TCP congestion control



Roadmap Transport Layer

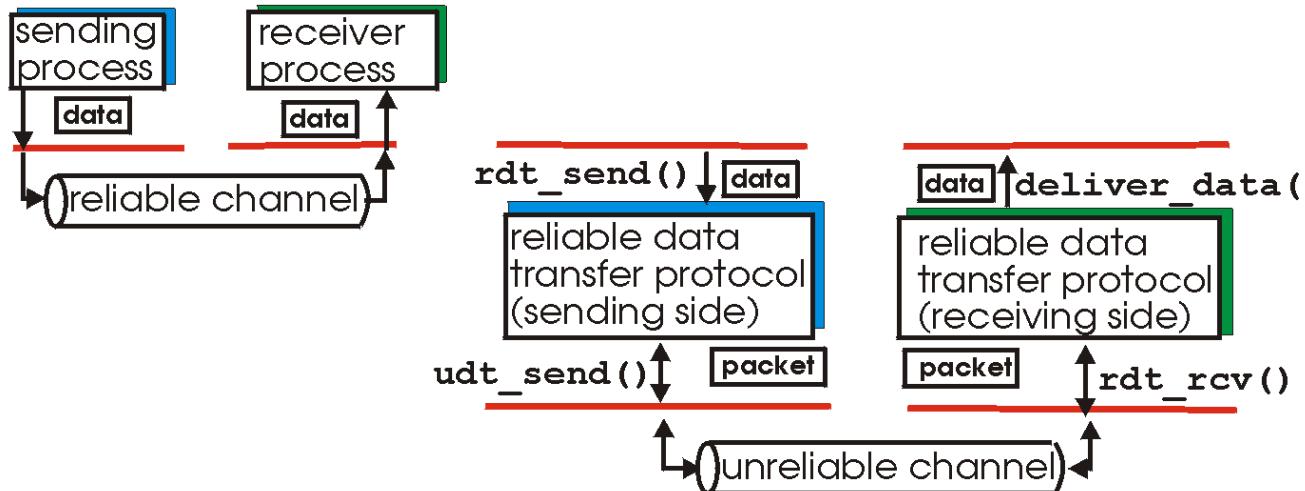


- transport layer services
- multiplexing/demultiplexing
- connectionless transport: UDP
- principles of reliable data transfer
 - Reliable transfer
 - Acknowledgements
 - Retransmissions
 - Connection management
 - Flow control and buffer space
 - Congestion control
 - Principles
 - TCP congestion control



Reliable data transfer (RDT) – Repetition

application layer
transport layer



Problem: We want to transfer data reliably over an unreliable channel

- Channel with bit errors
 - Underlying channel may flip bits in packets
 - But what if the ACK msgs. is corrupted?

- Channel loses packets
 - Underlying channel may randomly drop packets (data and ACKs)



- RDT 2.0
 - Error detection
 - Feedback: control msgs (ACK, NAK) from receiver to sender
 - Add sequence number, send multiple ACK.
- RDT 3.0
 - Retransmit after time-out
 - Handling of duplicates (ACK just delayed, not lost), stop & wait
 - Pipelining to increase throughput of data transfer.
 - Go-Back-n approach
 - Selective repeat

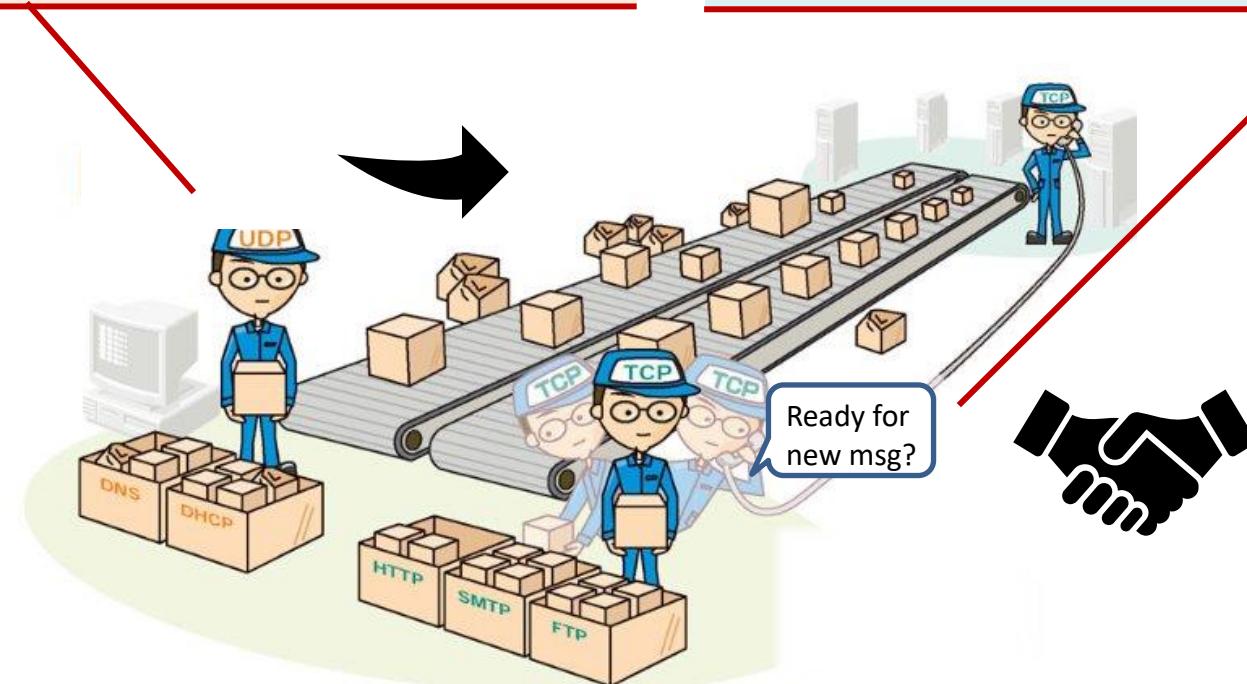
Services to upper layer by Internet transport protocols

UDP's service:

- *Connectionless: unreliable transport* between sending and receiving process
 - “best-effort” delivery
- *does not provide:* reliability, throughput, security guarantees

TCP's service:

- *connection-oriented: reliable transport* between sending and receiving process
 - correct, in-order data delivery, synch required between client, server
- *does not provide:* throughput, security guarantees



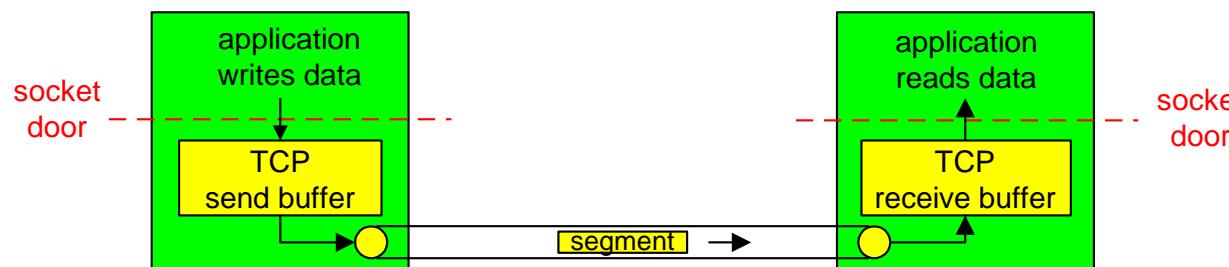
Roadmap Transport Layer

- transport layer services
- multiplexing/demultiplexing
- connectionless transport: UDP
- principles of reliable data transfer
- connection-oriented transport: TCP
 - Reliable transfer
 - Acknowledgements
 - Retransmissions
 - Connection management
 - Flow control and buffer space
 - Congestion control
 - Principles
 - TCP congestion control



TCP: Overview RFCs: 793,1122,1323, 2018, 5681, 7323

- point-to-point & full duplex data:
 - bi-directional data flow in same connection (S-R, R-S)
- connection-oriented, reliable, in-order byte sequence:
 - Needs handshaking (exchange of control msgs); ack-based; init sender & receiver state before data exchange; agree on Maximum Segment Size (MSS)
- Flow control
 - ack-based; sender will **not flood the receiver**
- (+ extra) Congestion control
 - sender will **not flood the network**



TCP segment structure

ACK: seq # of next expected
byte; a bit: this is an ACK

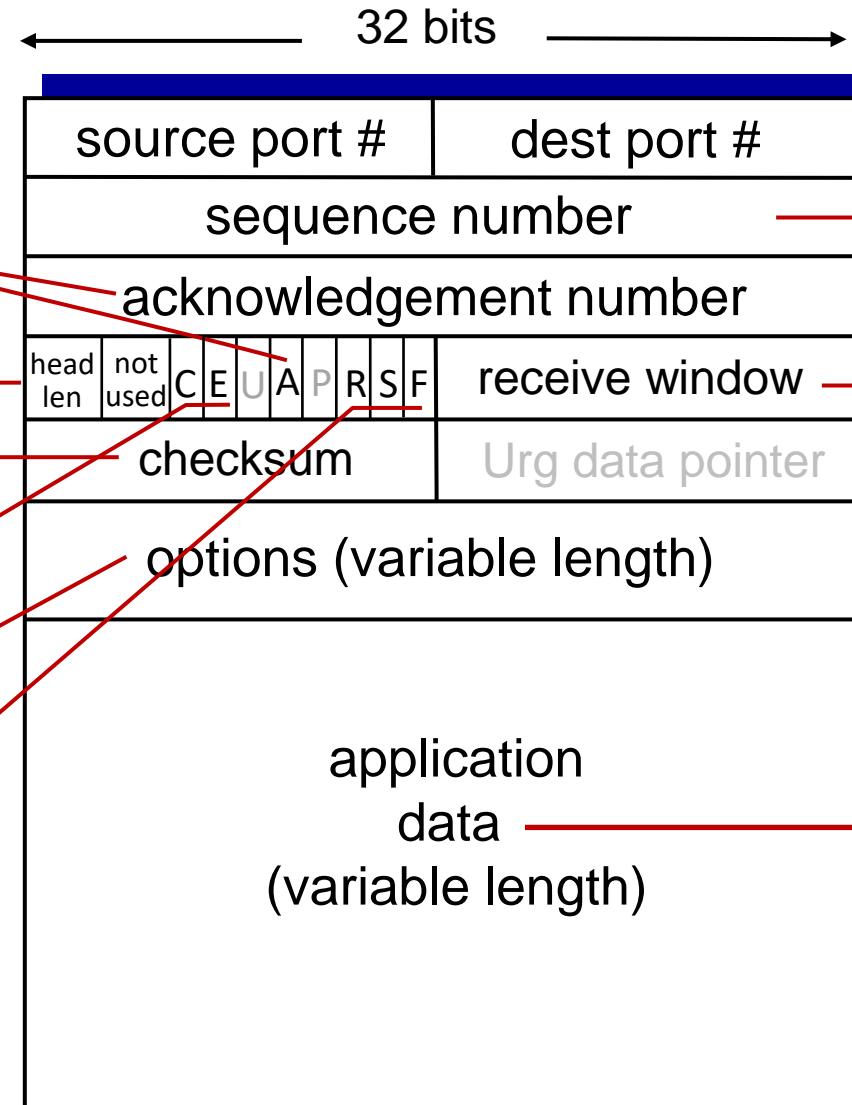
length (of TCP header)

Internet checksum

C, E: congestion notification

TCP options

RST, SYN, FIN: connection
management



segment seq #: counting
bytes of data into bytestream
(*not segments!*)

flow control: # bytes receiver willing to accept

data sent by application into TCP socket

Roadmap Transport Layer

- transport layer services
- multiplexing/demultiplexing
- connectionless transport: UDP
- principles of reliable data transfer
- connection-oriented transport: TCP
 - Reliable transfer
 - Acknowledgements
 - Retransmissions
 - Connection management
 - Flow control and buffer space
 - Congestion control
 - Principles
 - TCP congestion control



TCP seq. numbers, ACKs

sequence numbers:

- number of first **byte** in segment's data

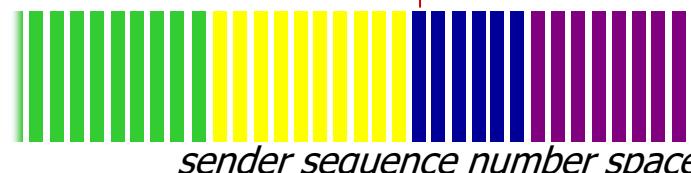
acknowledgements:

- seq # of **next in-order-byte expected**
- cumulative ACK**

outgoing segment from sender

source port #	dest port #
sequence number	
acknowledgement number	
	rwnd
checksum	

window size
 N



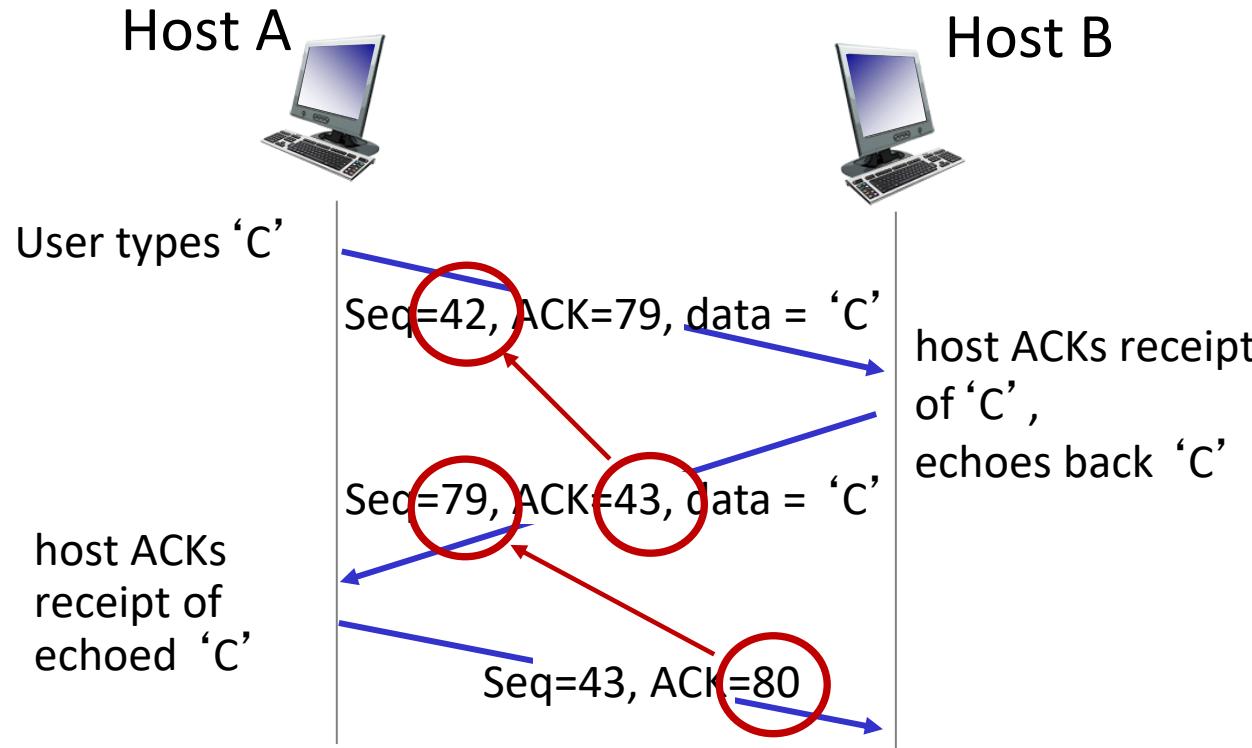
sent ACKed sent, not-yet ACKed ("in-flight") usable but not yet sent not usable

incoming segment to sender

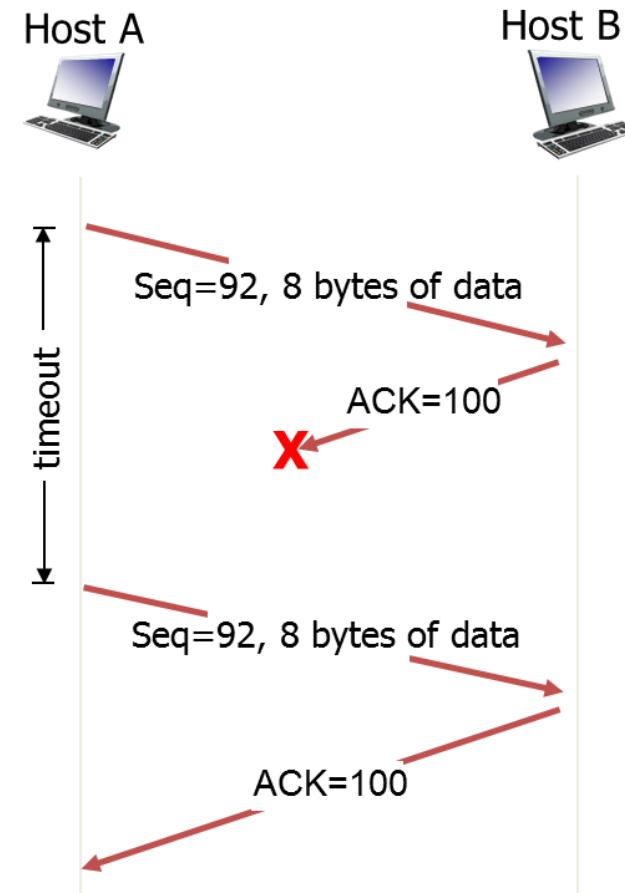
source port #	dest port #
sequence number	
acknowledgement number	
A	rwnd
checksum	

TCP seq. numbers, ACK - in action

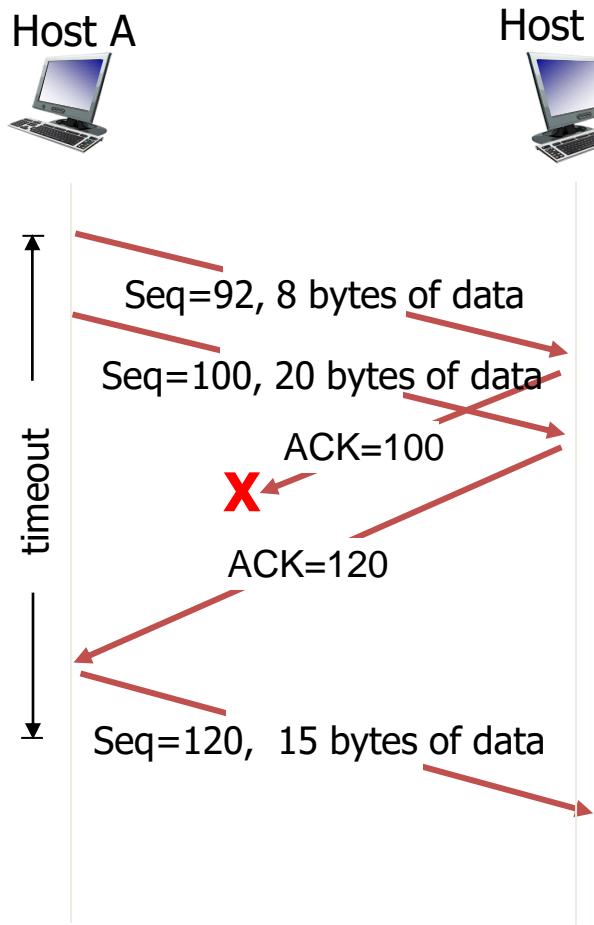
Always ack next in-order expected byte



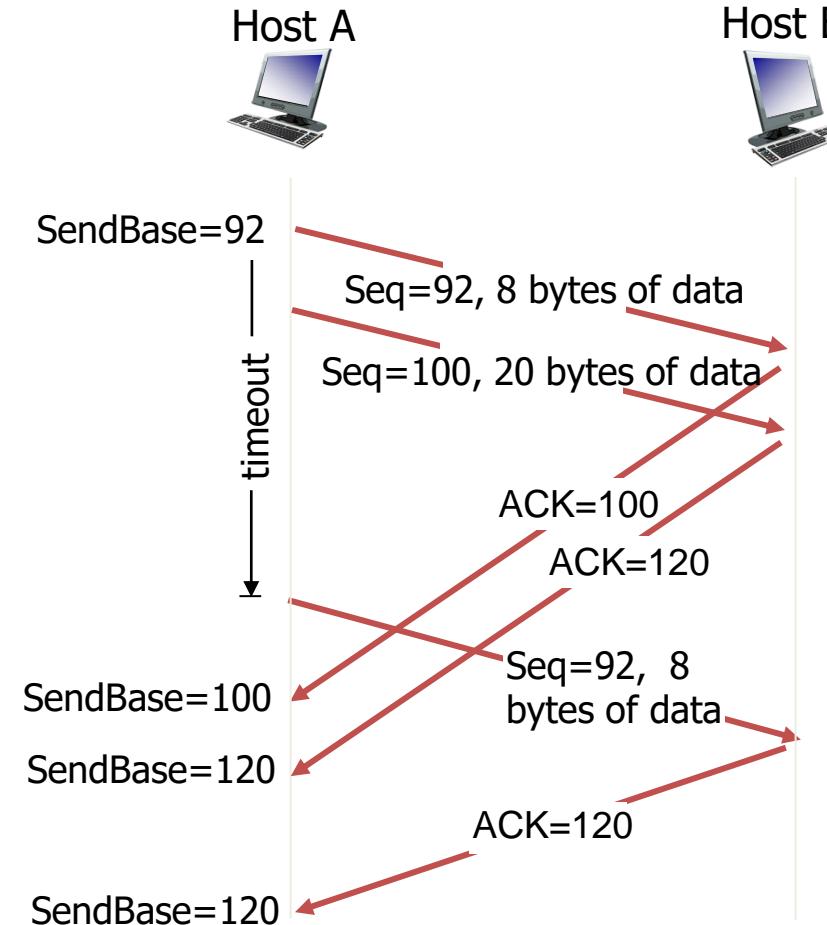
Simple example scenario
Based on telnet msg exchange



TCP: cumulative Ack - retransmission scenarios

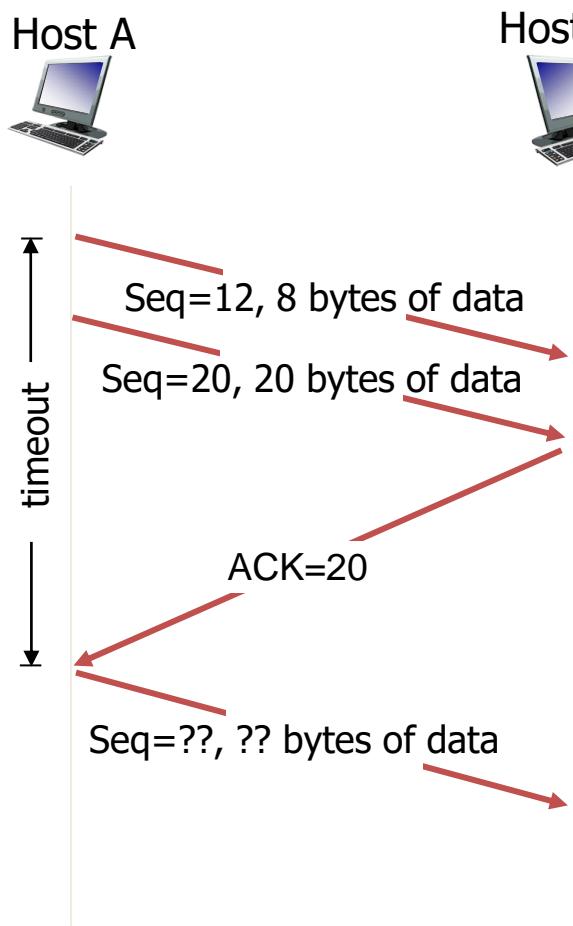


Cumulative ACK

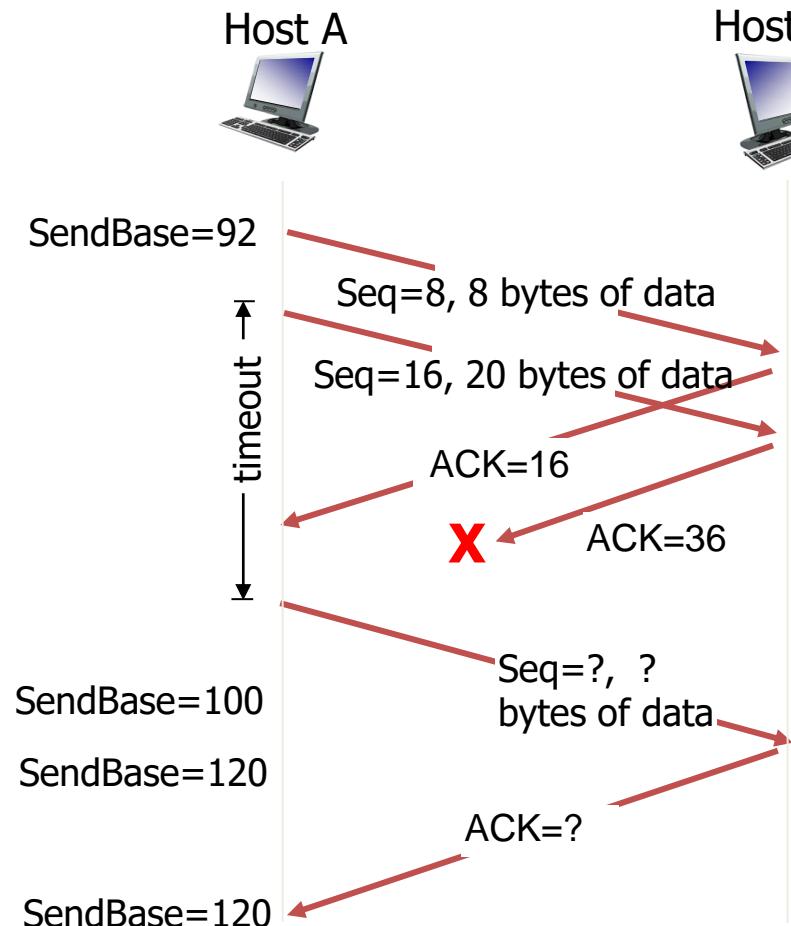


(Premature) timeout

TCP: cumulative Ack - retransmission scenarios



Cumulative ACK



(Premature) timeout

TCP ACK generation [RFC 1122, RFC 5681]

Event	TCP Receiver action
in-order segment arrival, no gaps, everything else already ACKed	Delayed ACK. Wait max 500ms for next segment then send ACK
in-order segment arrival, no gaps, one delayed ACK pending	immediately send single cumulative ACK
out-of-order segment arrival higher-than-expect seq. # gap detected	send (duplicate) ACK, indicating seq. # of next expected byte

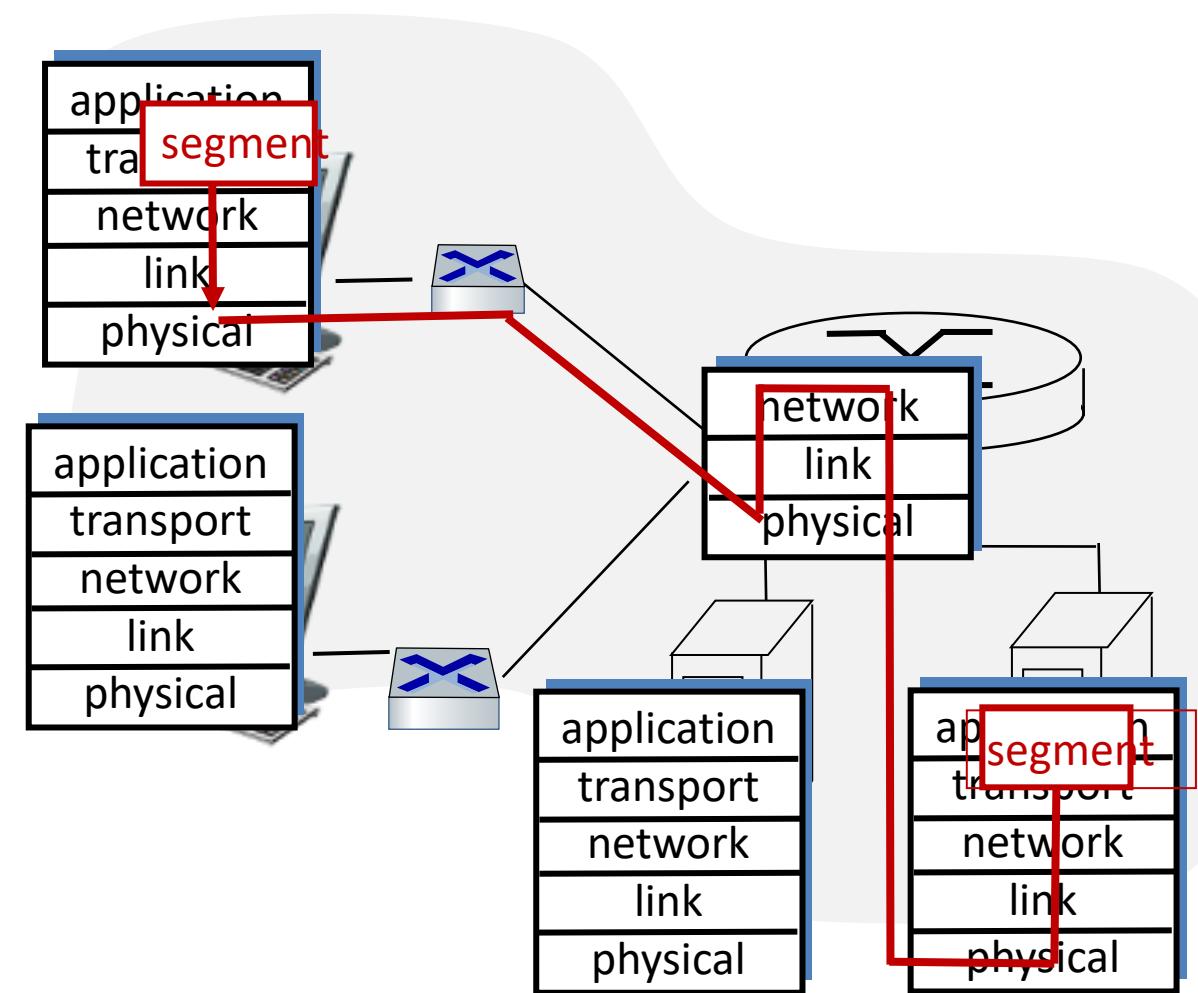
Roadmap Transport Layer

- transport layer services
- multiplexing/demultiplexing
- connectionless transport: UDP
- principles of reliable data transfer
- connection-oriented transport: TCP
 - Reliable transfer
 - Acknowledgements
 - Retransmissions
 - Connection management
 - Flow control and buffer space
 - Congestion control
 - Principles
 - TCP congestion control



Q: how to set TCP timeout value?

- longer than end-to-end RTT
 - but exact value varies!!!
- *too short timeout?*
 - premature, unnecessary retransmissions
- *too long?*
 - slow reaction to loss



TCP round trip time, timeout estimation

$$\text{EstimatedRTT} = (1-\alpha) * \text{EstimatedRTT} + \alpha * \text{SampleRTT}$$

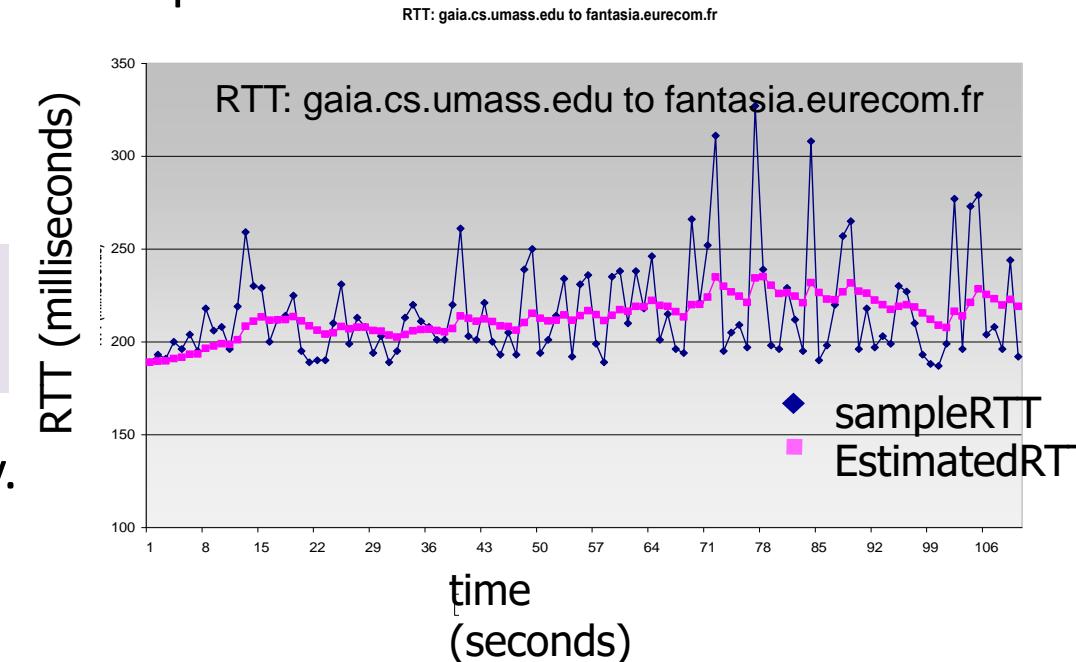
- exponential weighted moving **average**: influence of past sample decreases exponentially fast
- typical value: $\alpha = 0.125$

$$\text{DevRTT} = (1-\beta) * \text{DevRTT} + \beta * | \text{SampleRTT} - \text{EstimatedRTT} |$$

- Additionally, we also estimate a measure of variability.
- Typical value: $\beta = 0.25$

$$\text{TimeoutInterval} = \text{EstimatedRTT} + 4 * \text{DevRTT}$$

“safety margin”



TCP fast retransmit (RFC 5681)

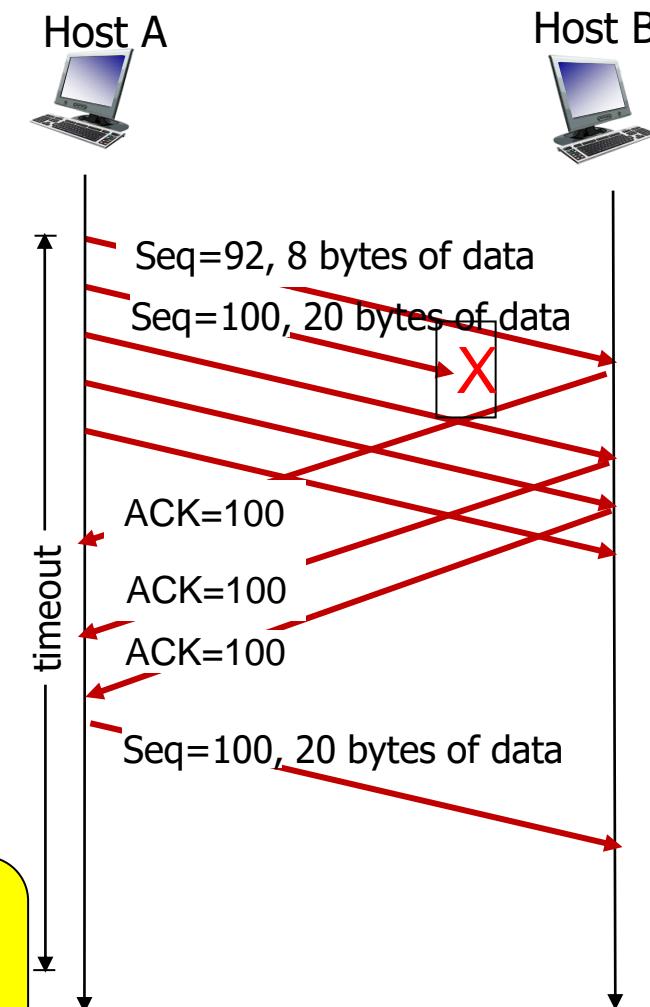
- time-out can still be long sometimes =>
 - long delay before resending lost data
- IMPROVEMENT: “suspect” loss via duplicate ACKs

TCP fast retransmit

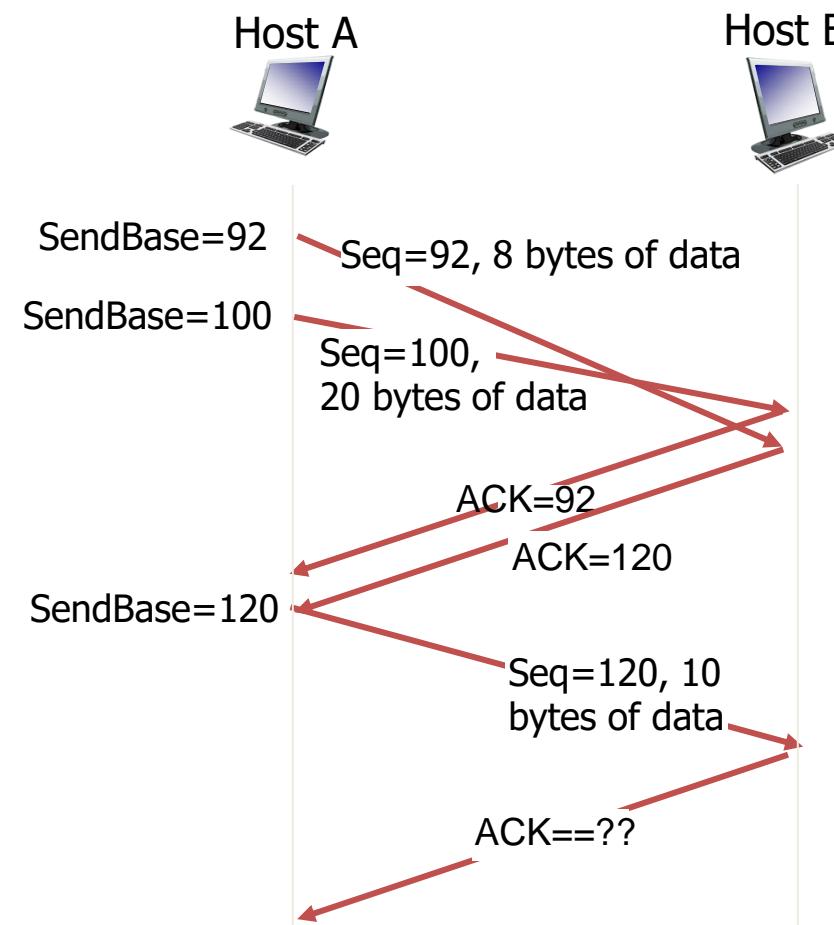
if sender receives 3 duplicate ACKs for same data

- likely that unacked segment lost, so don't wait for timeout

Implicit NACK!
Q: Why need at least 3?
(Hint: recall NW layer can “reorder” pkts)



TCP: segments can arrive in wrong order



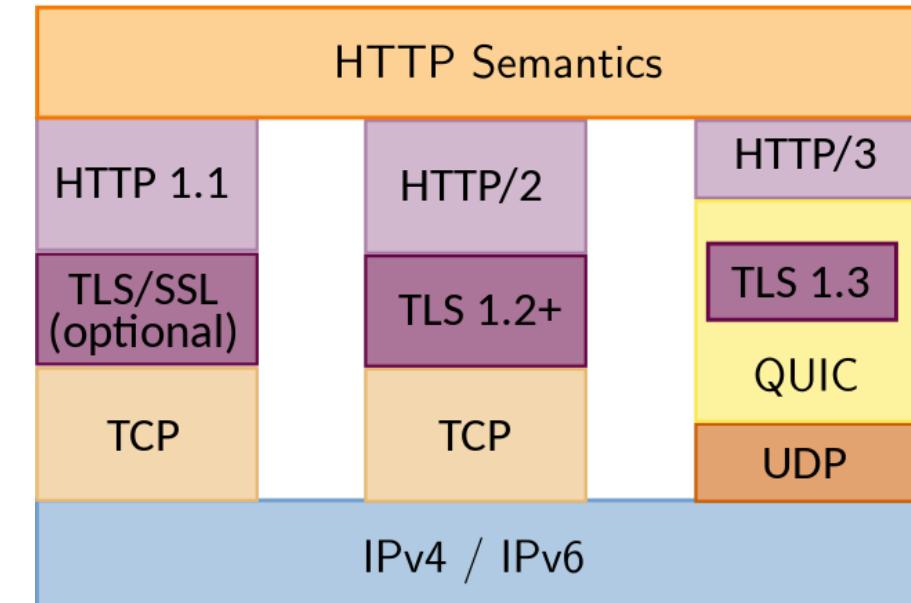
Is it possible to have reliable transfer over UDP?

Reliable transfer over UDP?

- Yes, add reliability at application
 - application-specific error recovery
 - more discussion in the context of multimedia applications and **evolving NW edge**



- HTTP/3 can use a new transport protocol called QUIC.
- QUIC builds on UDP and adds reliability (and security) on top.
- Why have TCP then?



Roadmap Transport Layer

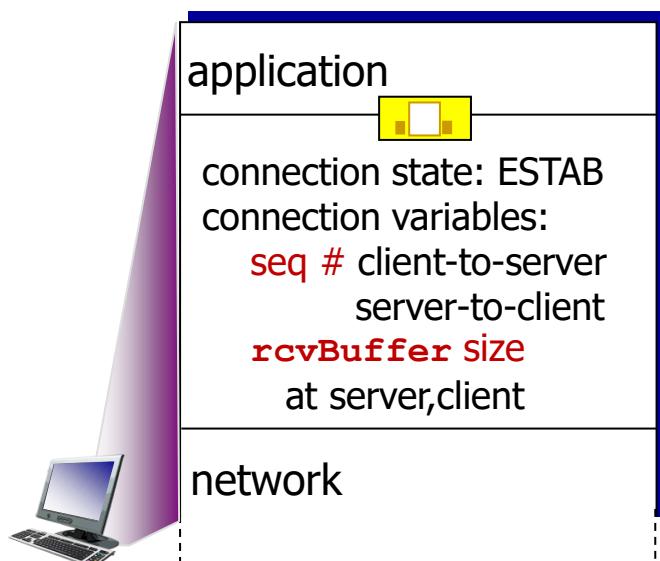
- transport layer services
- multiplexing/demultiplexing
- connectionless transport: UDP
- principles of reliable data transfer
- connection-oriented transport: TCP
 - Reliable transfer
 - Acknowledgements
 - Retransmissions
 - Connection management
 - Flow control and buffer space
 - Congestion control
 - Principles
 - TCP congestion control



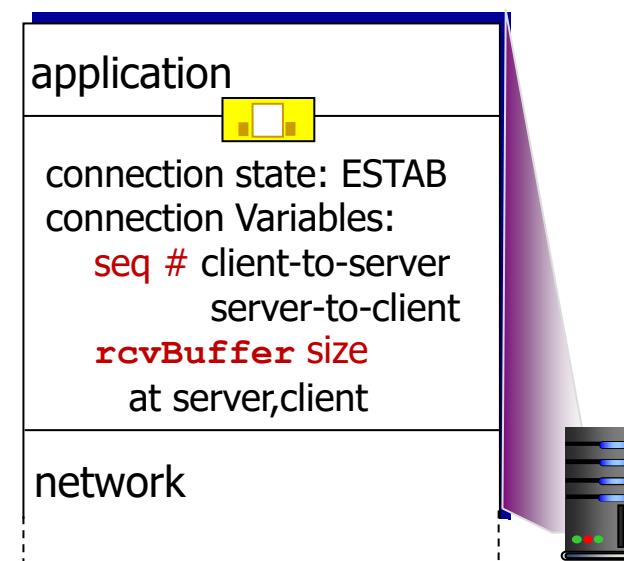
Connection Management

before exchanging data, sender/receiver “handshake”:

- agree to establish connection + decide connection parameters

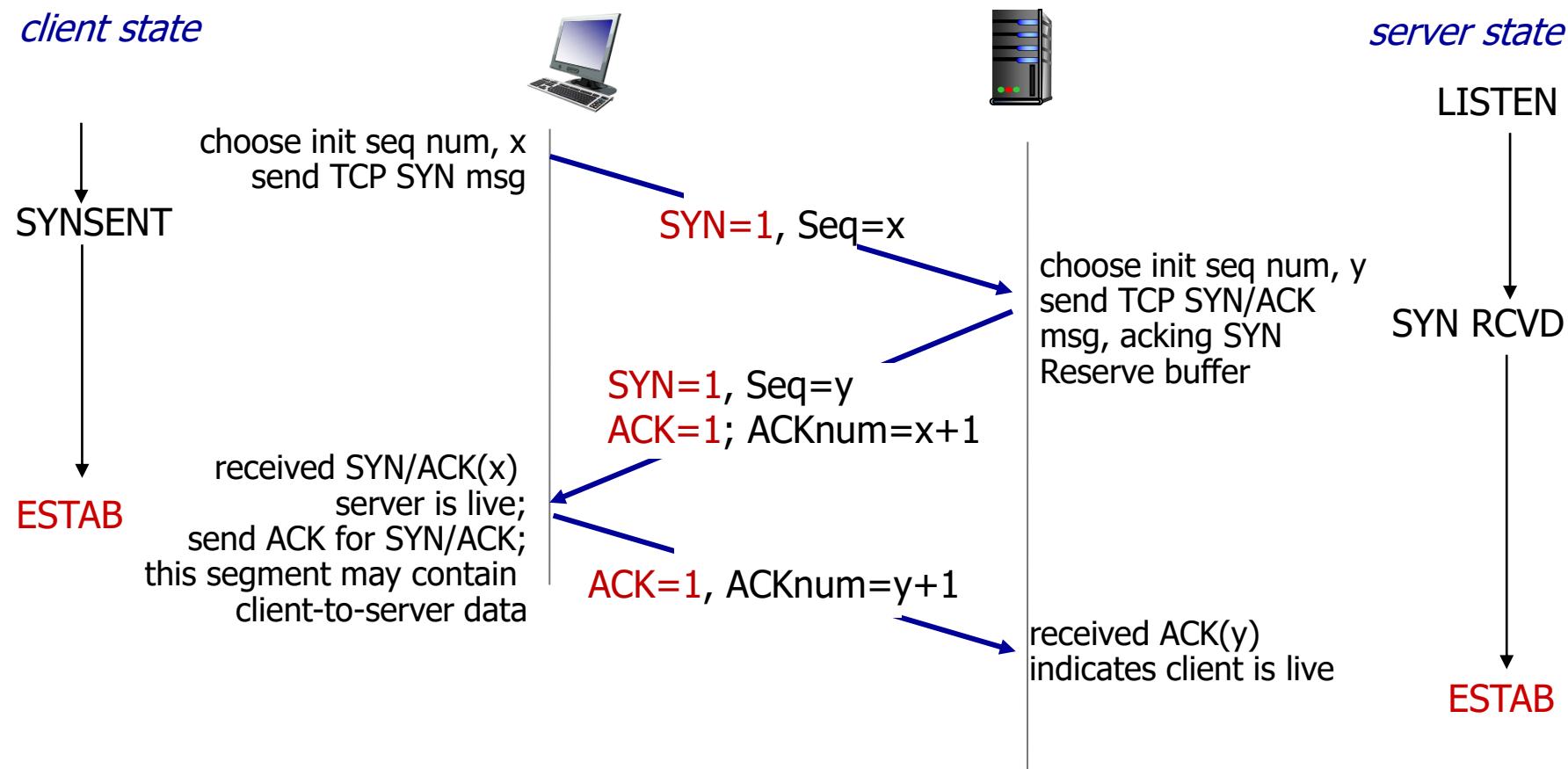


```
Socket clientSocket =  
    newSocket("hostname", "port  
    number");
```

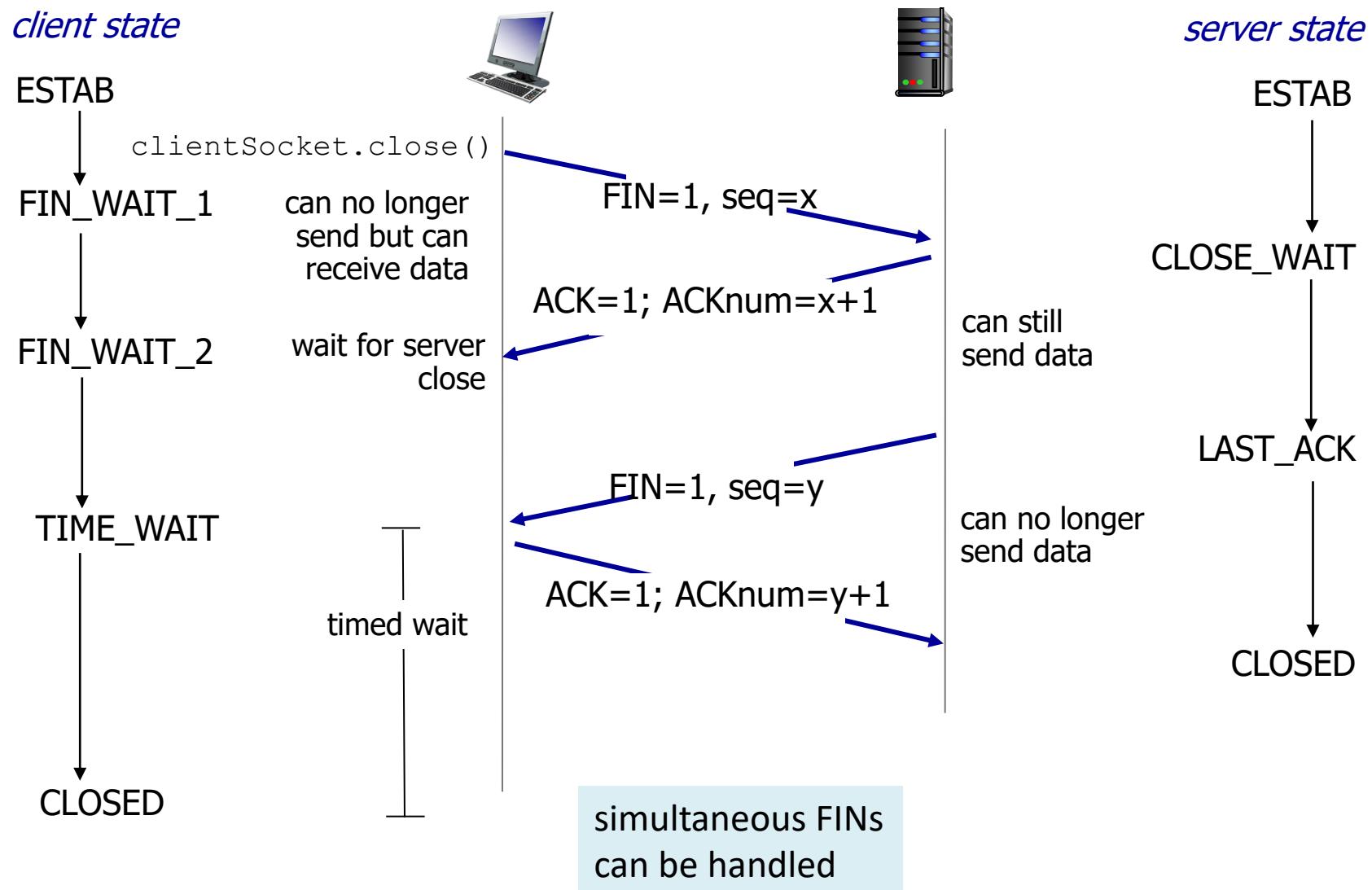


```
Socket connectionSocket =  
    welcomeSocket.accept();
```

Setting up a connection: TCP 3-way handshake

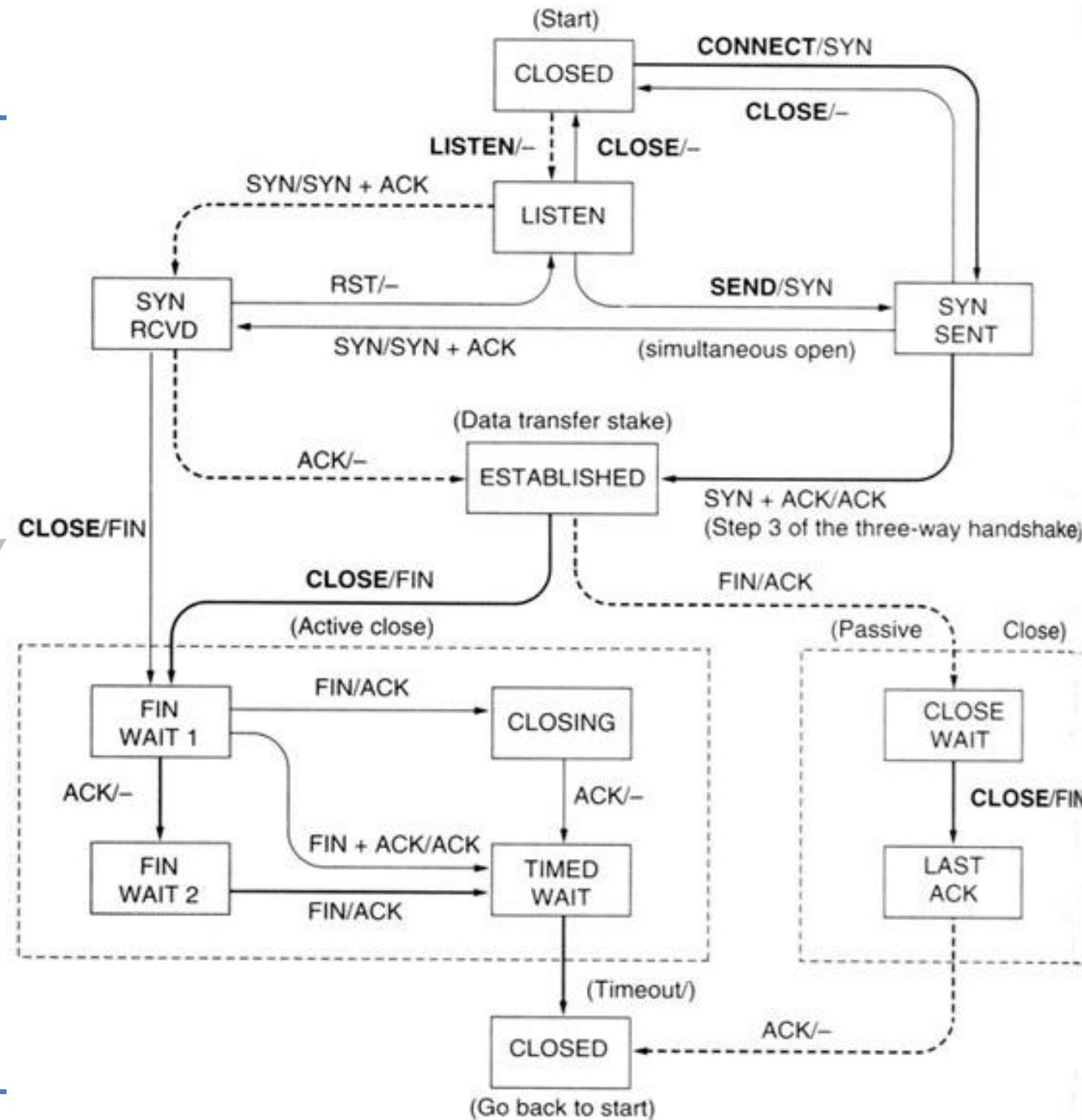


TCP: closing a connection



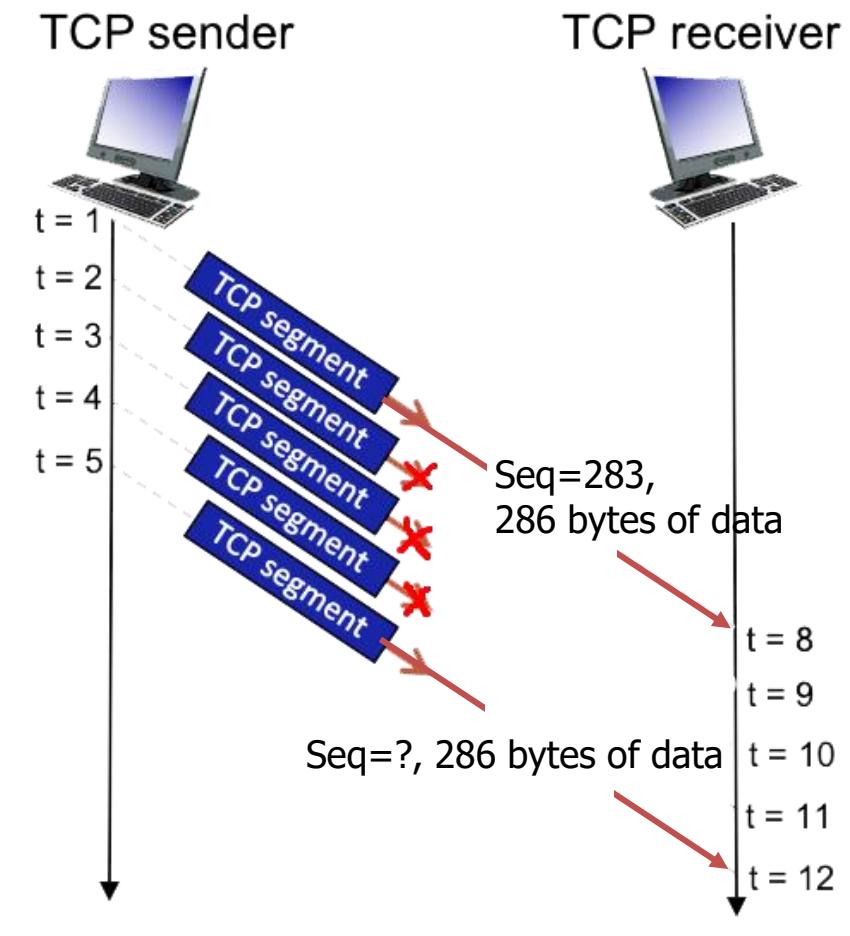
Part of the TCP state machine RFC-793 (setup and close states)

Legend:
event/action



Exercise

- What are the sequence number of each segment sent by sender?
- What are the ACKs numbers the receiver sends in response to each segment?



Roadmap Transport Layer

- transport layer services
- multiplexing/demultiplexing
- connectionless transport: UDP
- principles of reliable data transfer
- connection-oriented transport: TCP
 - Reliable transfer
 - Acknowledgements
 - Retransmissions
 - Connection management
 - Flow control and buffer space
 - Congestion control
 - Principles
 - TCP congestion control



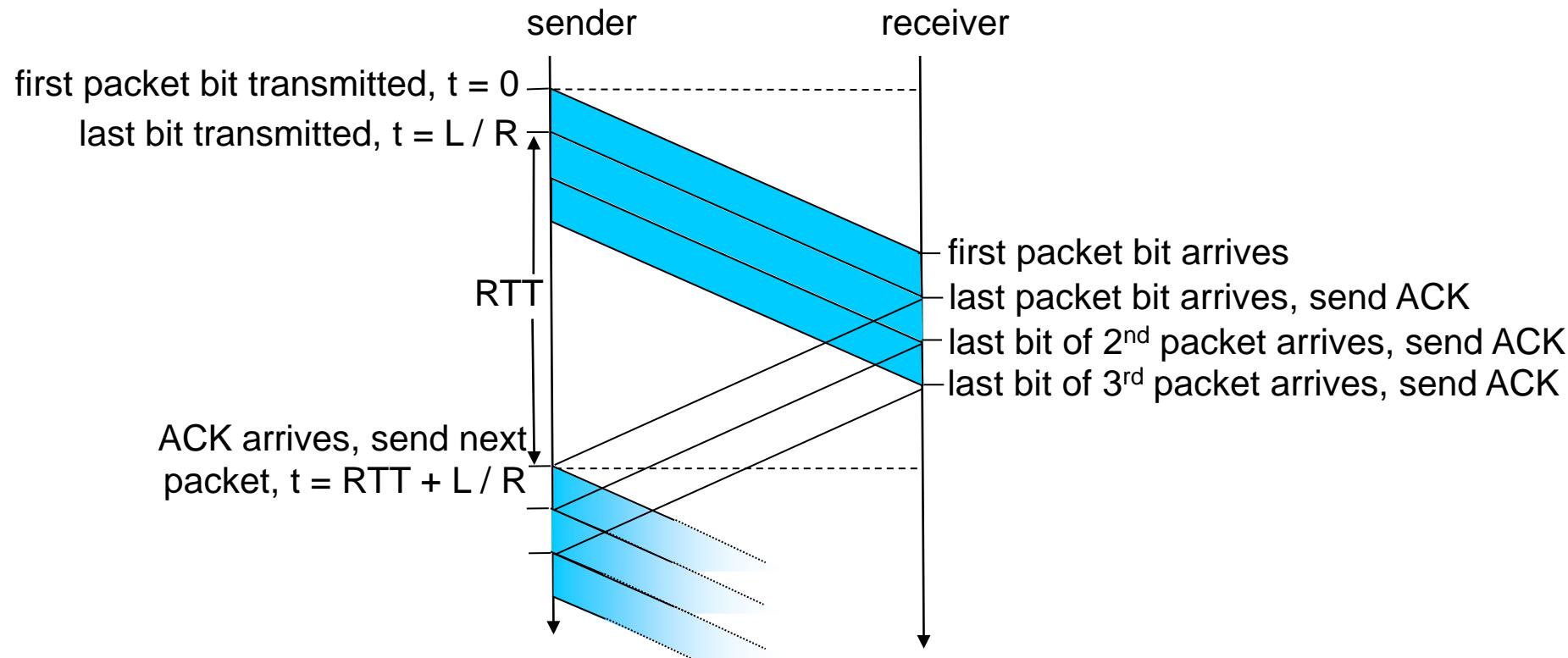
Flow control: what is the problem to address?



Have we seen something that helps to deal with this?
ACK's can help for this too (besides RDT)

This Photo by Unknown Author is licensed under [CC BY-SA](#)

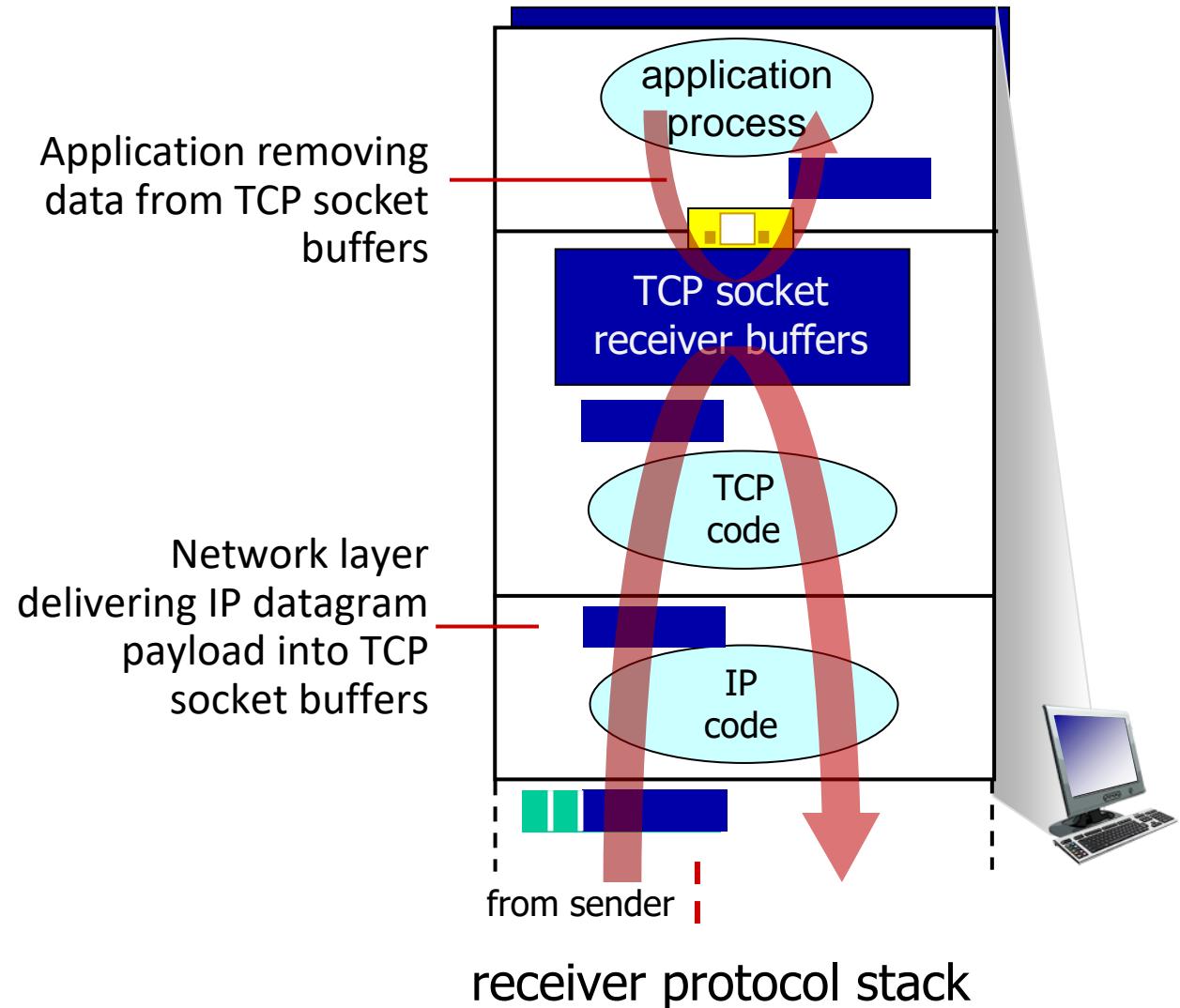
Ack-based pipelining => error-control & flow control at the same time!!!



Flow control: Sender/Receiver problem; Sender cares to not overwhelm Receiver

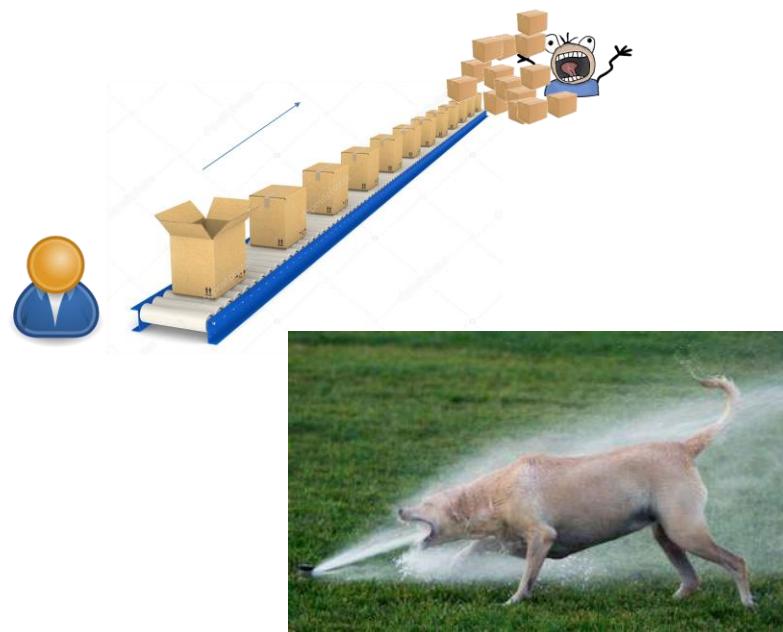
TCP flow control

Q: What happens if network layer delivers data faster than application layer removes data from socket buffers?



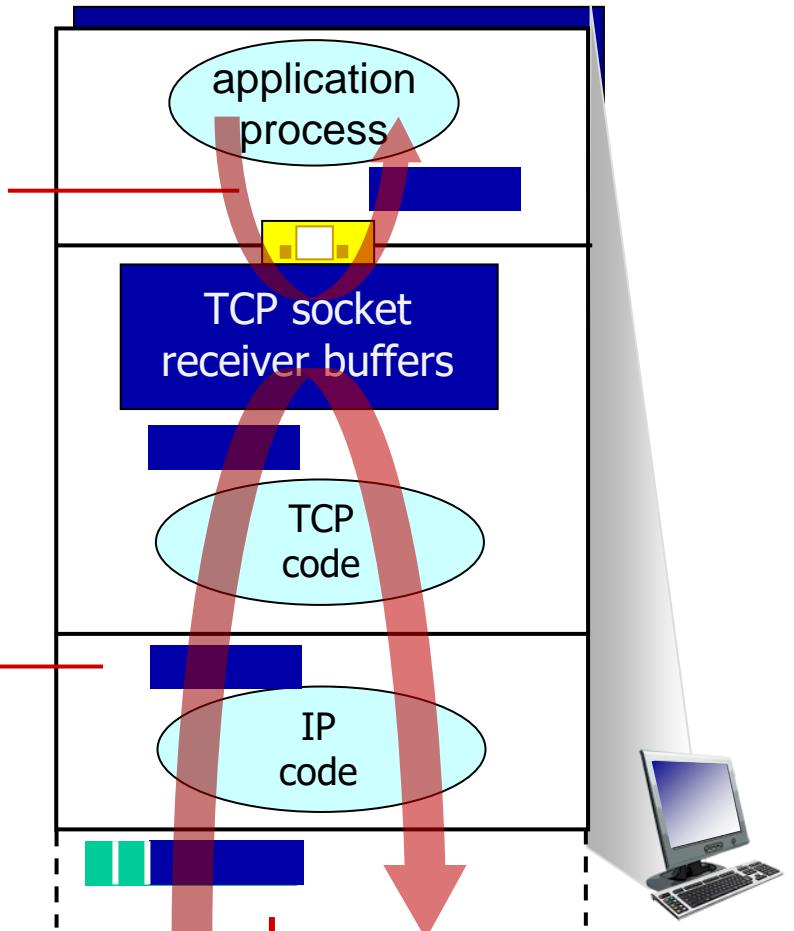
TCP flow control

Q: What happens if network layer delivers data faster than application layer removes data from socket buffers?



Application removing data from TCP socket buffers

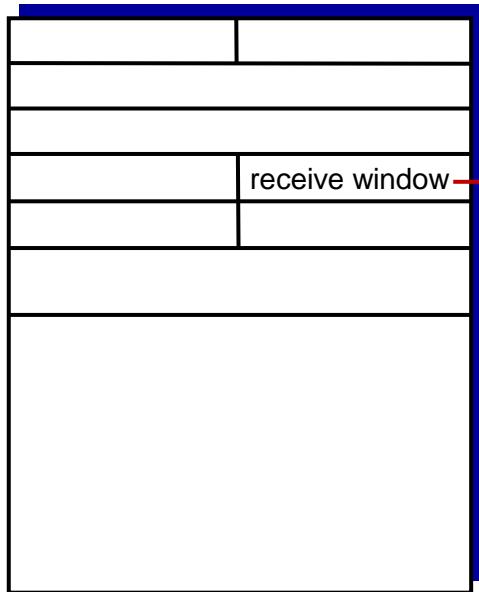
Network layer delivering IP datagram payload into TCP socket buffers



receiver protocol stack

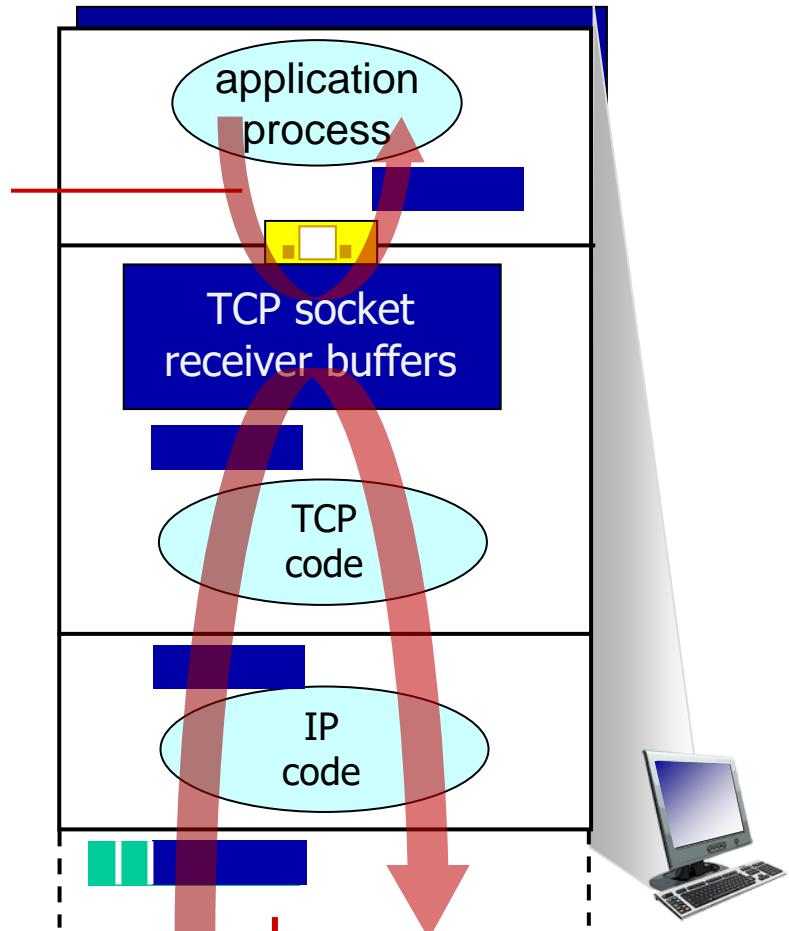
TCP flow control

Q: What happens if network layer delivers data faster than application layer removes data from socket buffers?



flow control: # bytes receiver willing to accept

Application removing data from TCP socket buffers



receiver protocol stack

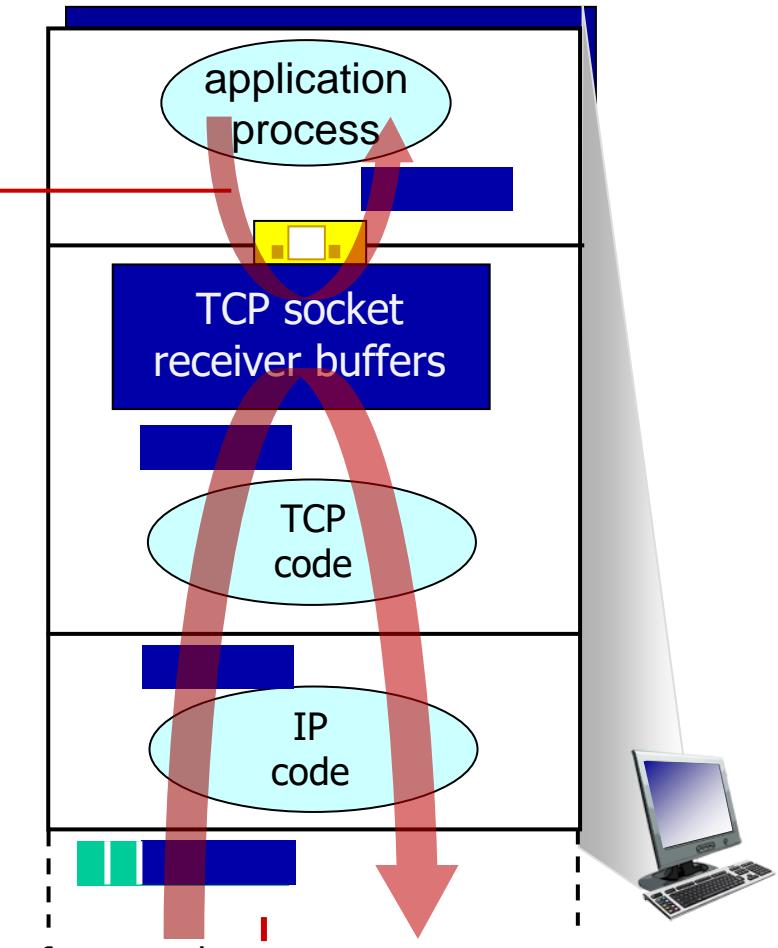
TCP flow control

Q: What happens if network layer delivers data faster than application layer removes data from socket buffers?

flow control

receiver controls sender, so sender won't overflow receiver's buffer by transmitting too much, too fast

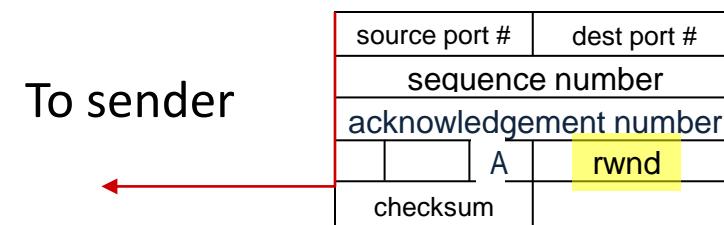
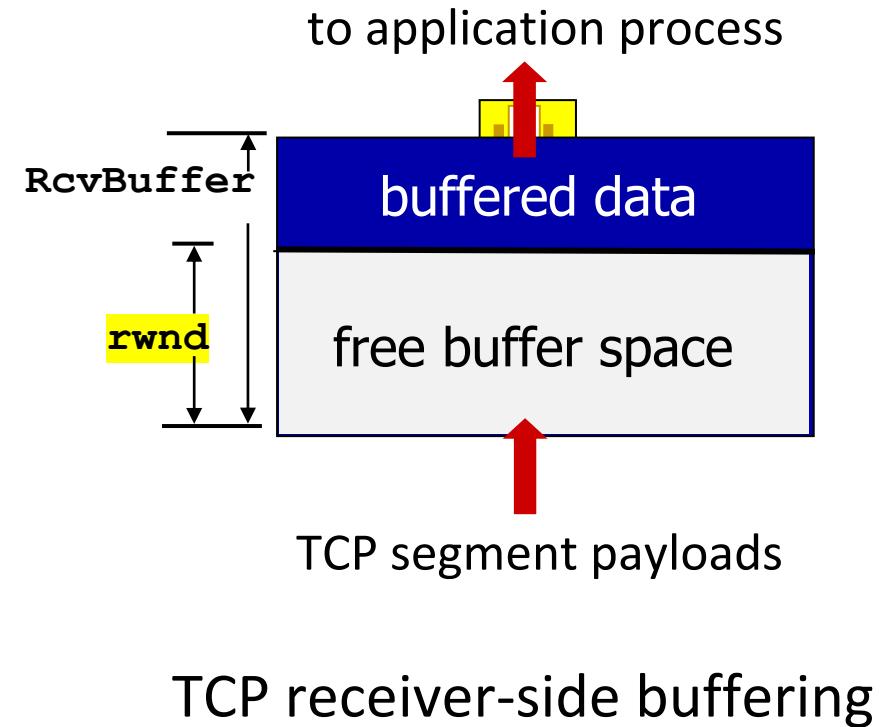
Application removing data from TCP socket buffers



receiver protocol stack

TCP flow control

- receiver “advertisises” free buffer space through **rwnd** value in header
 - **RcvBuffer** size set via socket options
 - OS can autoadjust **RcvBuffer**
- sender limits unacked (“in-flight”) data to receiver’s **rwnd** value
 - s.t. receiver’s buffer will not overflow

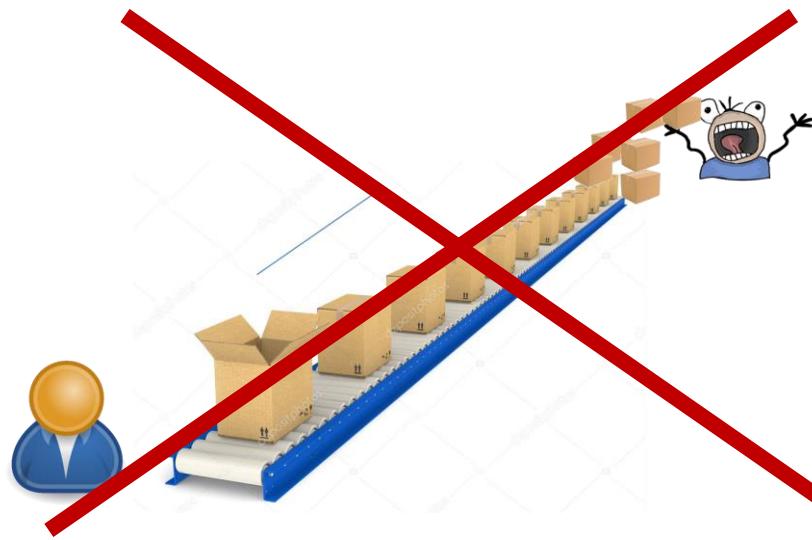


Roadmap Transport Layer

- transport layer services
- multiplexing/demultiplexing
- connectionless transport: UDP
- principles of reliable data transfer
- connection-oriented transport: TCP
 - Reliable transfer
 - Acknowledgements
 - Retransmissions
 - Connection management
 - Flow control and buffer space
 - Congestion control
 - Principles
 - TCP congestion control



Q: Difference between flow control and congestion control?



Flow-control = *sender avoids swamping the receiver
(i.e. network edge) issue*



Congestion control = *Avoid congesting the network
(i.e. NW core issue)*

Distinction between flow control and congestion control

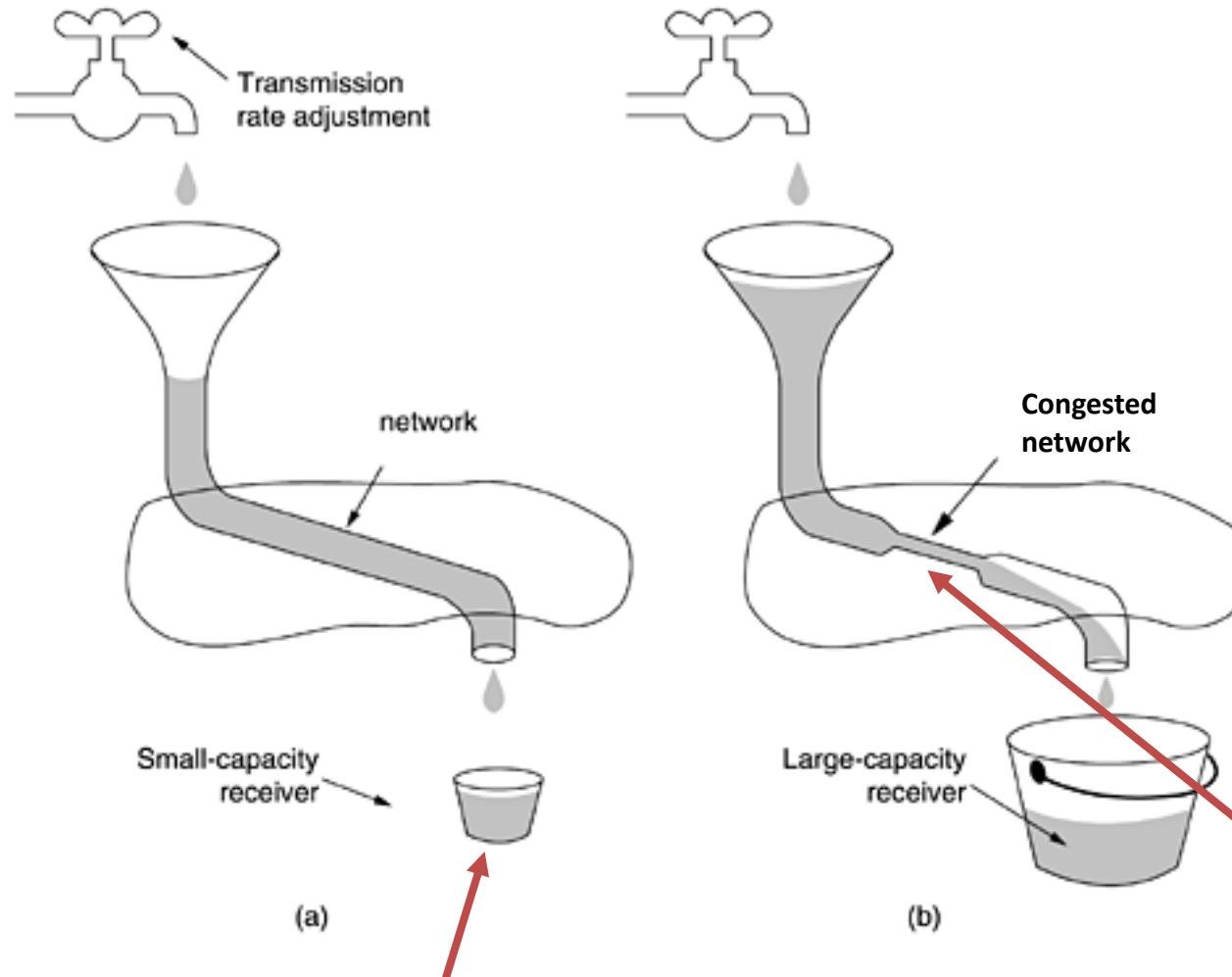


Fig. A. Tanenbaum
Computer Networks

Flow control deals with

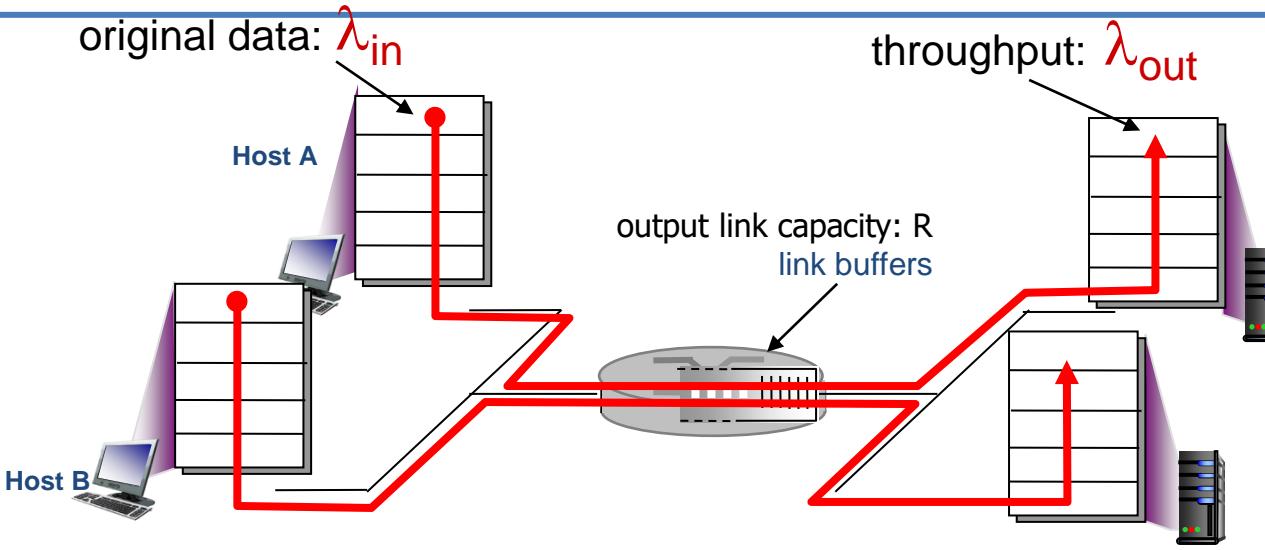
Congestion control deals with

About congestion

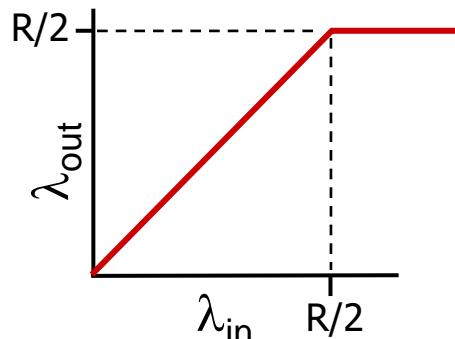
- Cause?
 - many sources sending too much data too fast for **network core** to handle
- Manifestations **@NW edge?**
 - lost packets (due to buffer overflow at routers)
 - long delays (due to queueing in router buffers)



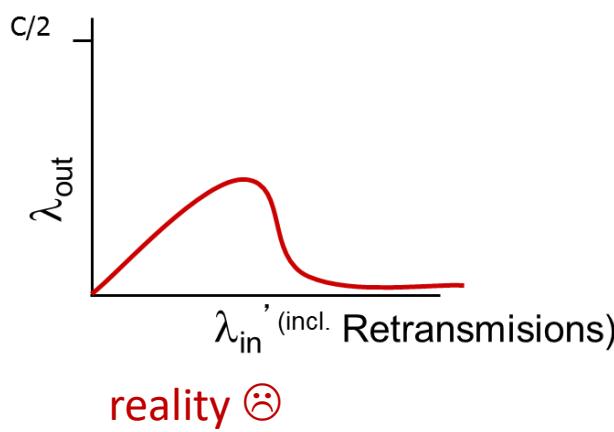
Causes/costs of congestion



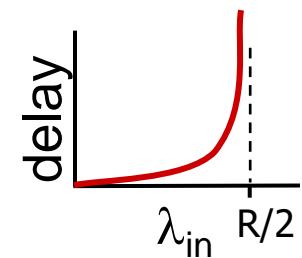
- Recall queueing behaviour + losses
- Losses => retransmissions => even higher load...



Ideal per-connection throughput:
 $R/2$ (if 2 connections)



reality ☺



Approaches towards congestion control

classic TCP

end-end congestion control:

- no explicit feedback from network
- congestion inferred from end-system observed behaviour

network-assisted congestion control:

- routers collaborate for optimal rates + provide feedback to end-systems eg.
 - indicate congestion
 - define explicit rate for sender to send at

Roadmap Transport Layer

- transport layer services
- multiplexing/demultiplexing
- connectionless transport: UDP
- principles of reliable data transfer
- connection-oriented transport: TCP
 - Reliable transfer
 - Acknowledgements
 - Retransmissions
 - Connection management
 - Flow control and buffer space
 - Congestion control
 - Principles
 - TCP congestion control



Q: How does TCP know if there is congestion in the network core?

Classic TCP congestion control: AIMD

- *approach:* senders can increase sending rate until **packet loss (congestion)** occurs; decrease sending rate on loss event

Additive Increase

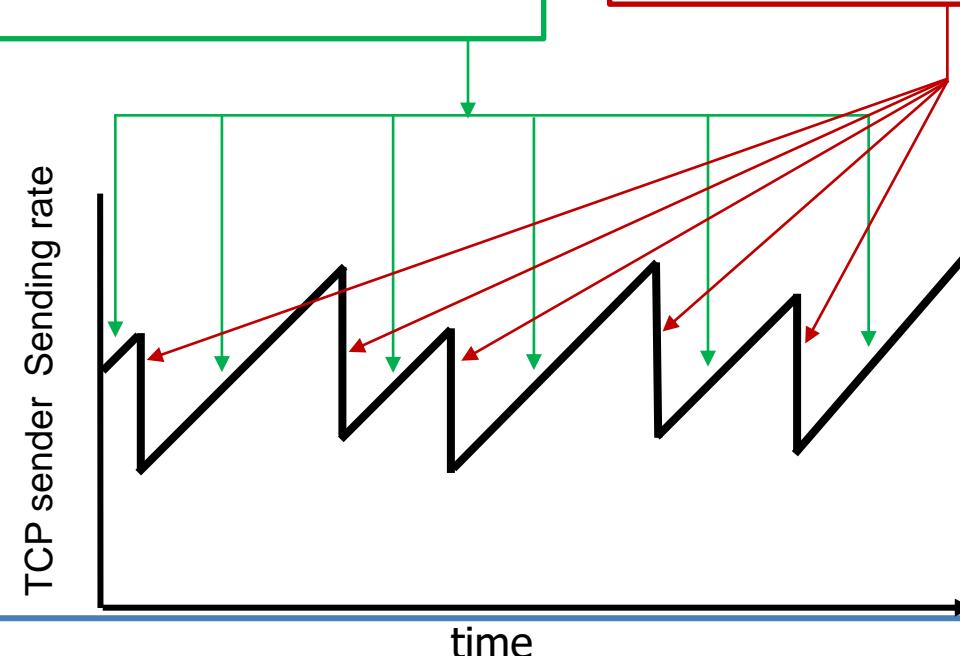
increase sending rate by 1 maximum segment size every RTT, until loss detected

Multiplicative Decrease

cut sending rate in half at each loss event

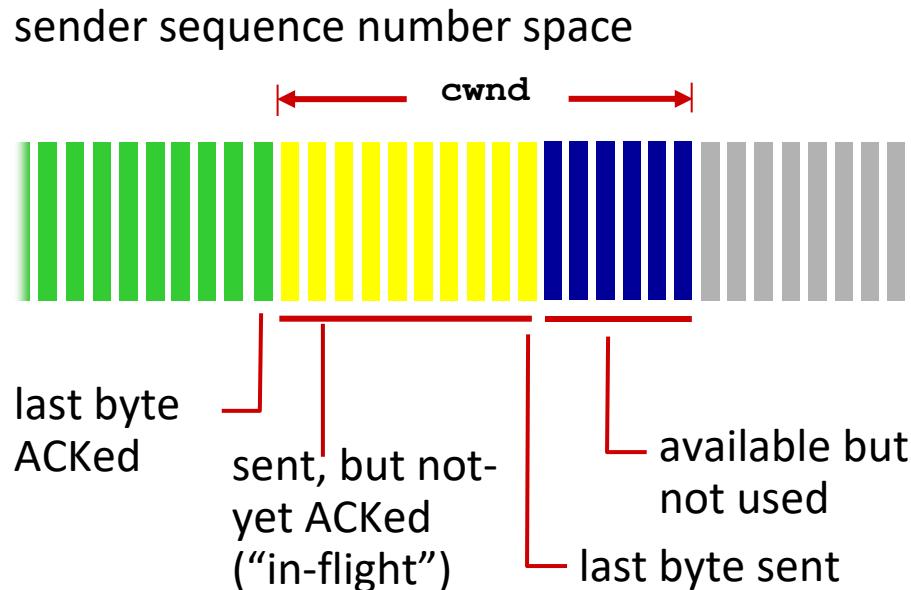
Why AIMD?

- distributed, asynchronous algorithm, shown to:
 - optimize rates network-wide!
 - have desirable stability properties



AIMD sawtooth behavior: *probing* for bandwidth

TCP congestion control: details



TCP sending behavior:

- *roughly*: send $cwnd$ bytes, wait RTT for ACKS, then send more bytes

$$\text{TCP rate} \approx \frac{cwnd}{RTT} \text{ bytes/sec}$$

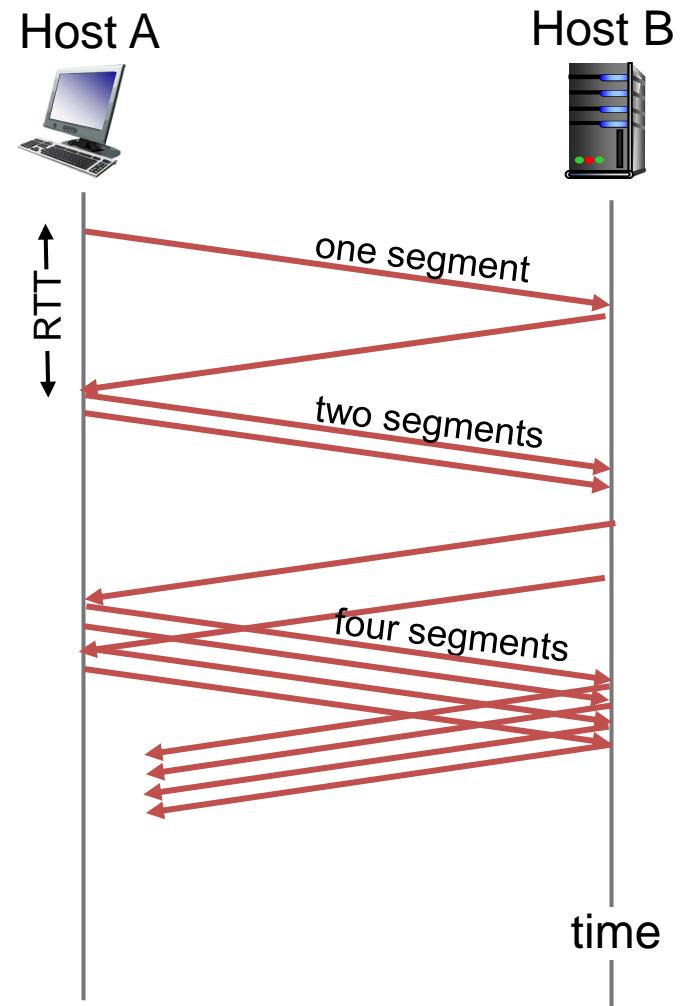
- TCP sender limits transmission: $\text{LastByteSent} - \text{LastByteAcked} \leq cwnd$
- Cwnd: dynamically adjusted in response to observed network congestion

TCP cwnd : Slow Start

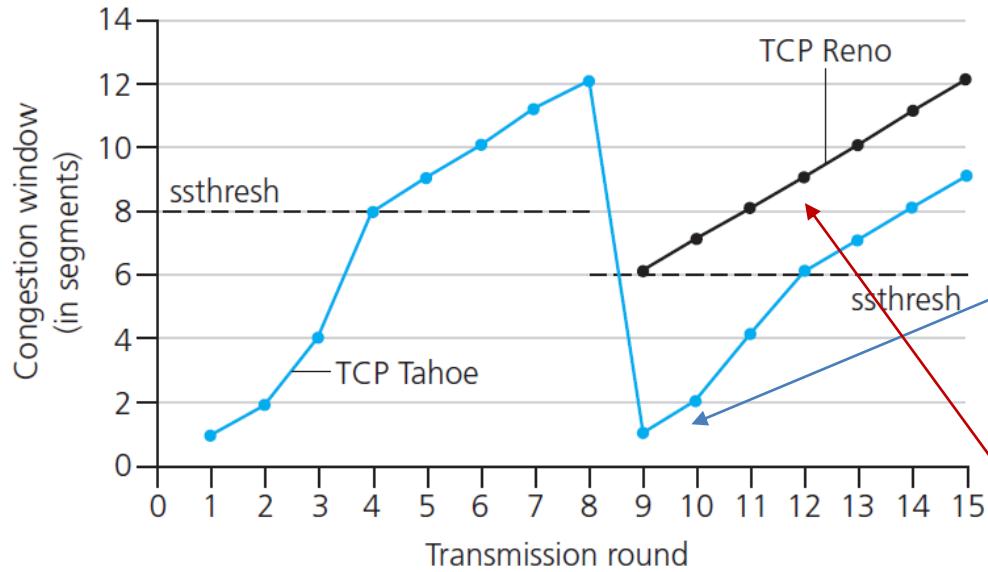
- when connection begins, increase rate exponentially until first loss event:
 - initially **cwnd** = 1 MSS
 - double **cwnd** every ack of previous “batch”
 - done by incrementing **cwnd** for every ACK received
 - until a **threshold**, then linear increase

Summary:

- initial rate slow but ramps up exponentially fast
- then, saw-tooth



TCP cwnd: from exponential to linear growth + reacting to loss



Tahoe: loss indicated by **timeout** or

3 duplicate ACKs:

enter **slow-start** phase: **cwnd** := 1 MSS and
exponential increase to **threshold (*)**, then enter
congestion avoidance phase (grow linearly)

(*) variable **ssthresh** (slow start threshold)

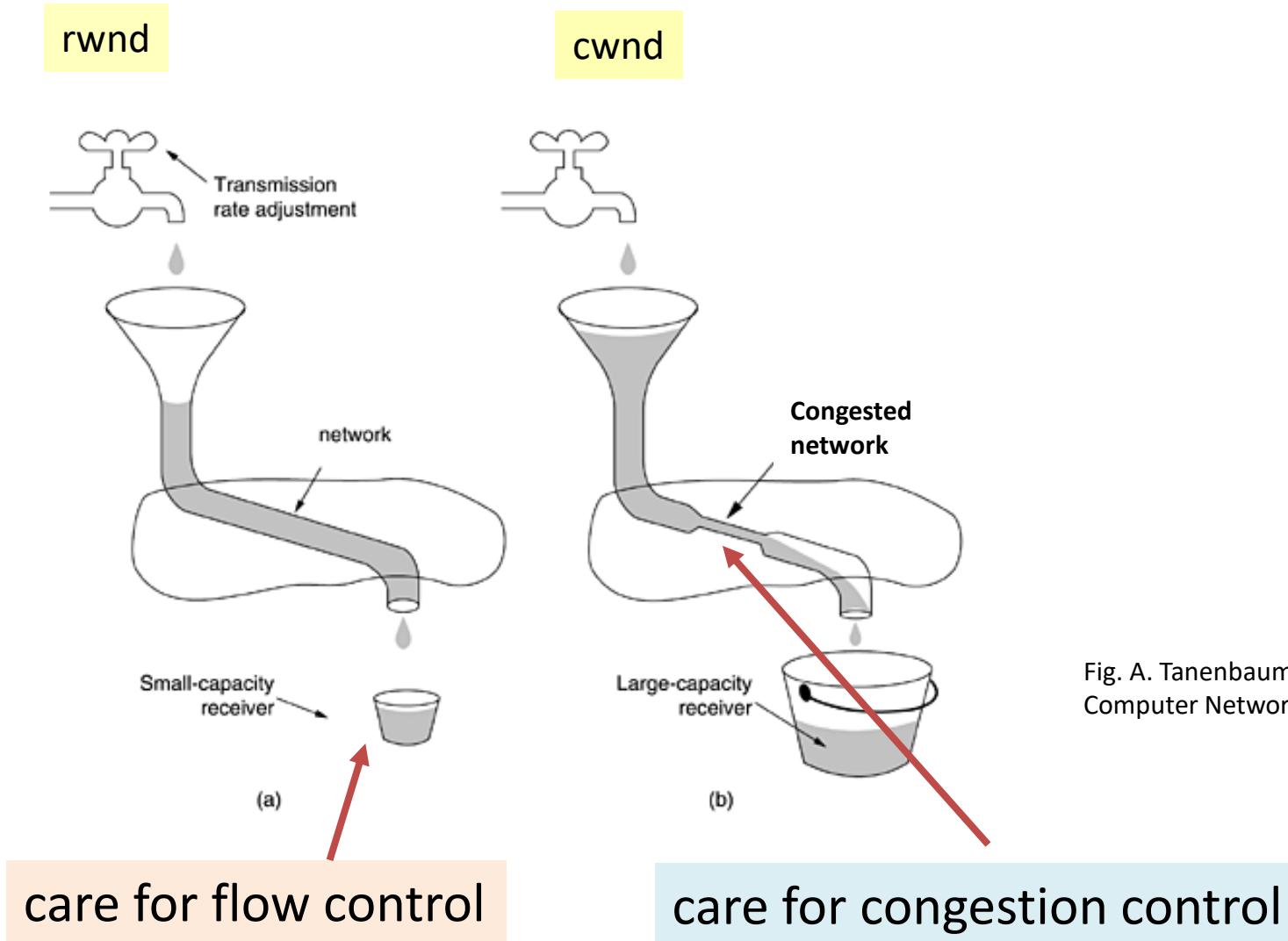
on loss event, $ssthresh = \frac{1}{2} * cwnd$

Reno (improved):

timeout: same as Tahoe (enter slow-start phase)

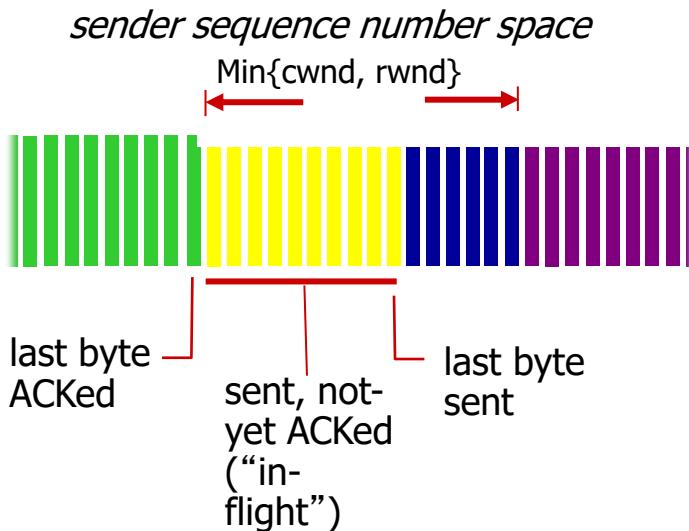
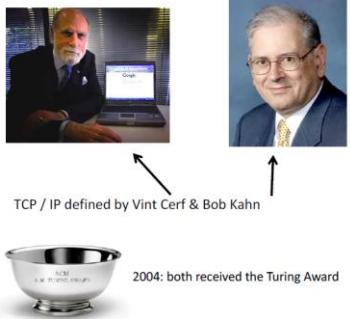
3 duplicate ACKs: cut **cwnd** in half and enter
congestion avoidance phase. Called **Fast Recovery**

TCP: 2 problems, joint solution: limit the rate of the sender! (or "How many window-variables does a TCP's sender maintain?")



TCP combined flow-ctrl, congestion ctrl windows

1974



TCP sending rate:

- send min {cwnd, rwnd} bytes, wait for ACKS, then send more

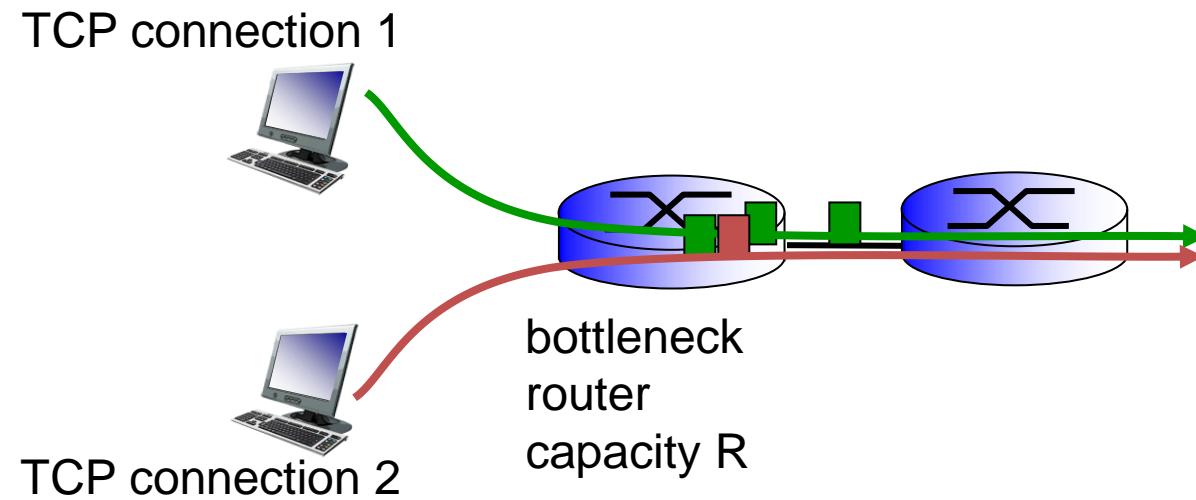
- sender limits transmission:

$$\frac{\text{LastByteSent} - \text{LastByteAcked}}{\text{LastByteAcked}} \leq \text{Min}\{\text{cwnd}, \text{rwnd}\}$$

- **cwnd** is dynamic, function of perceived network congestion
- **rwnd** dynamically limited by receiver's buffer space

TCP Fairness

fairness goal: if K TCP sessions share same bottleneck link of bandwidth R , each should have average rate of R/K



Q: Can a TCP *implementation* deviate from the Congestion-Control standard?

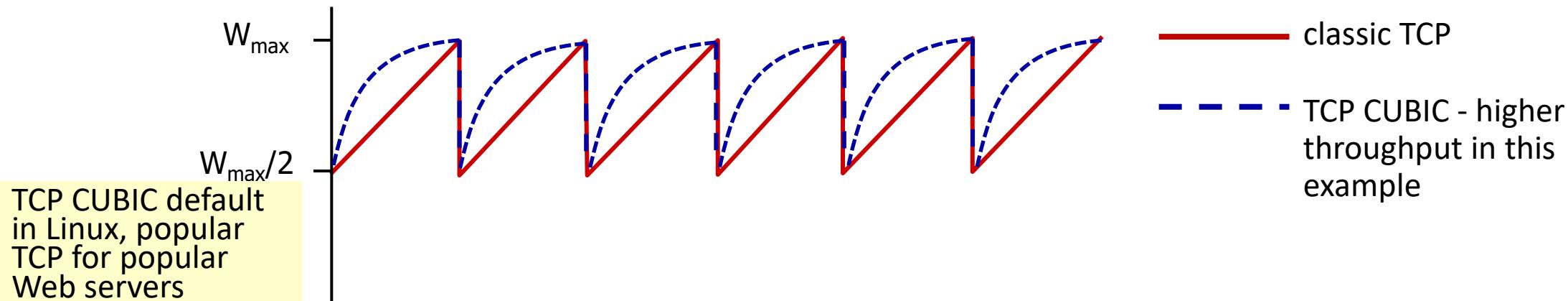
Roadmap Transport Layer

- transport layer services
- multiplexing/demultiplexing
- connectionless transport: UDP
- principles of reliable data transfer
- connection-oriented transport: TCP
 - reliable transfer
 - Acknowledgements
 - Retransmissions
 - Connection management
 - Flow control and buffer space
 - Congestion control
 - Principles
 - TCP congestion control (CC)
 - Encore/the (almost) last “mile”in TCP (CC and more)



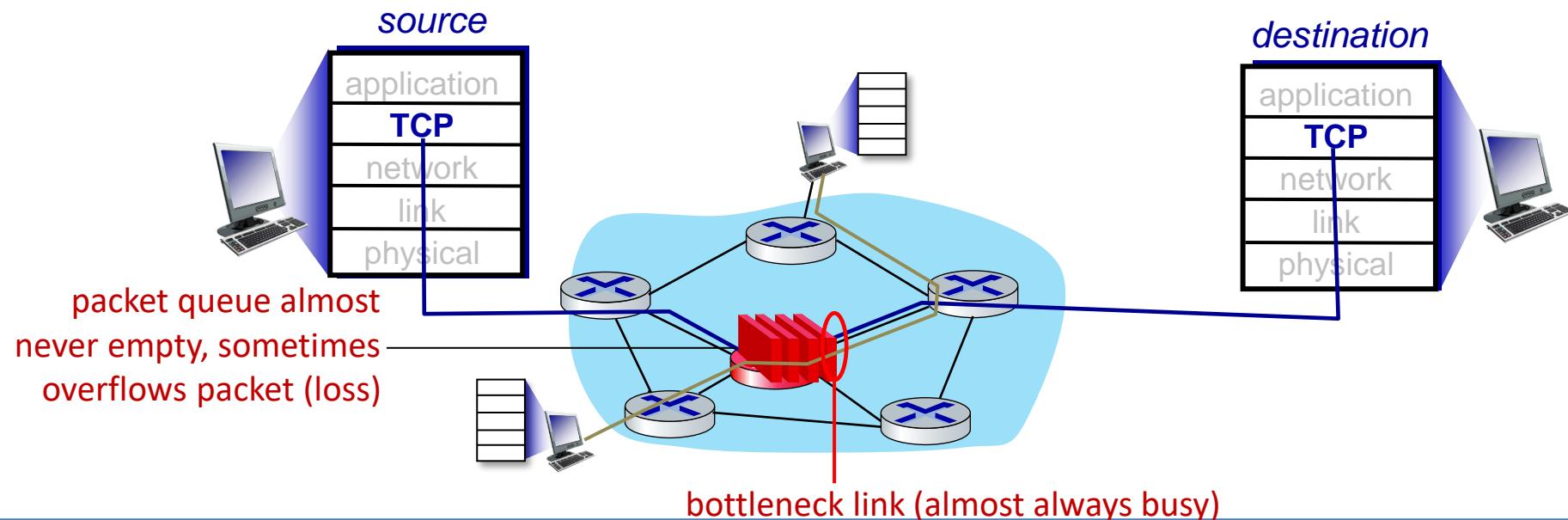
TCP CUBIC

- Insight/intuition:
 - W_{\max} : sending rate at which congestion loss was detected
 - after cutting rate in half on loss, initially ramp up to W_{\max} *faster*, but then approach W_{\max} more *slowly*
- increase W as a function of the *cube* of the distance between current time and K = estimated time to reach W_{\max}
 - larger increases when further away from K
 - smaller increases (cautious) when nearer K



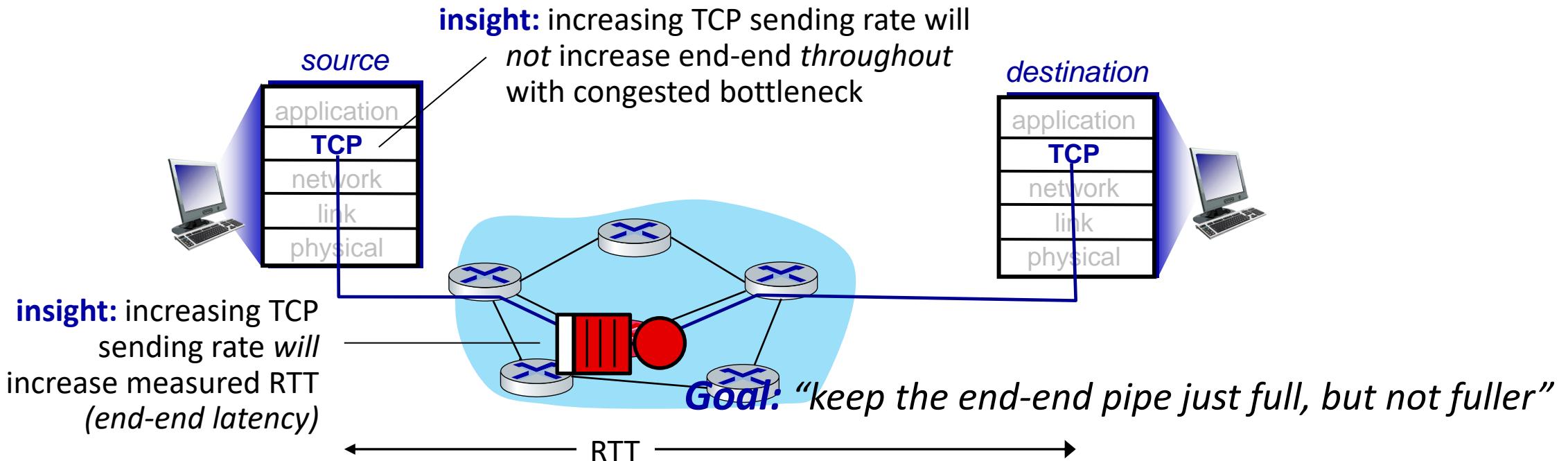
TCP and the congested “bottleneck link”

- TCP (classic, CUBIC) increase TCP's sending rate until packet loss occurs at some router's output: the *bottleneck link*



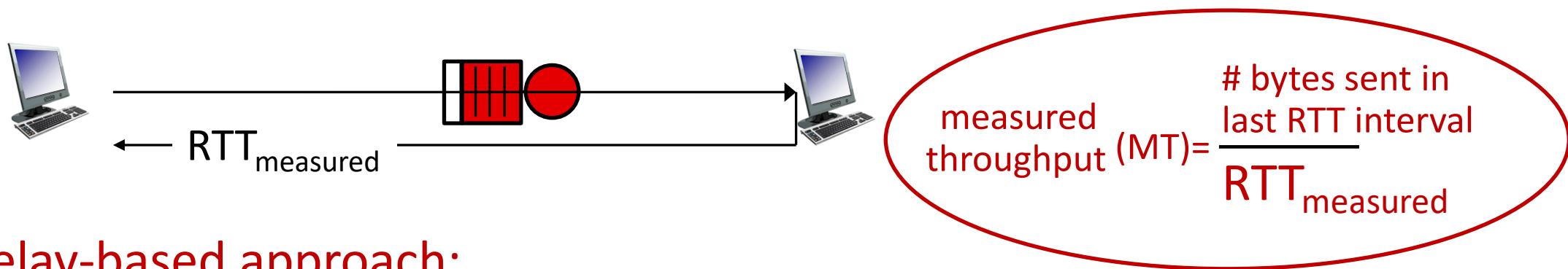
TCP and the congested “bottleneck link”

- TCP (classic, CUBIC) increase TCP's sending rate until packet loss occurs at some router's output: the *bottleneck link*



New: Delay-based TCP congestion control

Keep sender-to-receiver pipe “just full enough, but no fuller”: keep bottleneck link busy, but avoid high delays/buffering



Delay-based approach:

■ ~~RTT_{min} - minimum observed RTT (uncongested path)~~

■ ~~Estimated Uncongested Throughput (UT) = cwnd/RTT_{min}~~

if MT ~ UT : increase cwnd linearly

if MT << UT decrease cwnd linearly

/* path not congested */

/* path is congested */

BBR (Bottleneck Bandwidth and Round-trip propagation time):
deployed on Google's (internal)
backbone network
Cf: Cardwell, Neal, et al. "BBR
congestion-based congestion
control." Communications
ACM 2017

“spoiler”: Evolving transport-layer functionality

- TCP, UDP: principal transport protocols for 40 years
- TCP is not “problem-free”: *many different “flavors” of TCP developed*, for various scenarios:

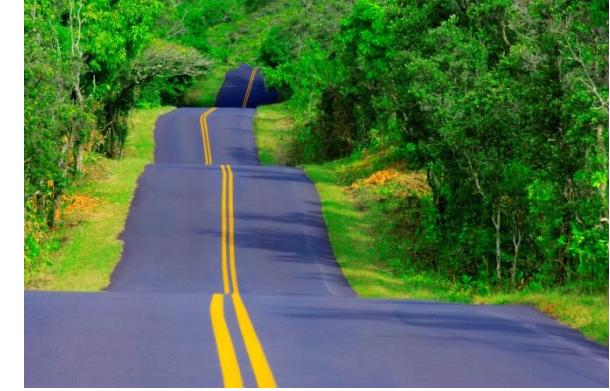
Scenario	Challenges
Long, fat pipes (large data transfers)	Many packets “in flight”; loss shuts down pipeline
Wireless networks	Loss due to noisy wireless links, mobility; TCP treat this as congestion
Long-delay links	Extremely long RTTs
Data center networks	Latency sensitive
Background traffic flows	Low priority, “background” TCP flows

- moving transport-layer functions to application layer, on top of UDP
 - HTTP/3: QUIC

(we will touch upon this issue in the cross-layer lectures on evolving Internet)

Roadmap Transport Layer

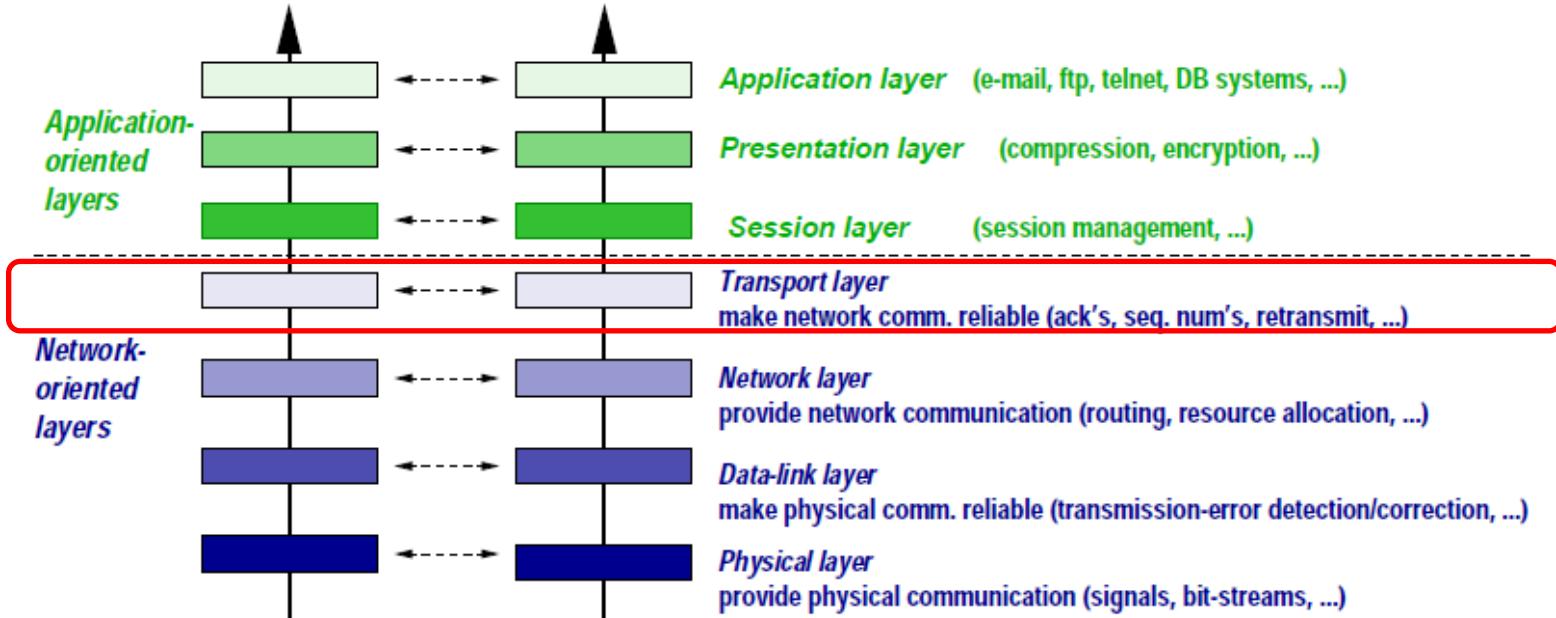
- transport layer services
- multiplexing/demultiplexing
- connectionless transport: UDP
- principles of reliable data transfer
- connection-oriented transport: TCP
 - reliable transfer
 - Acknowledgements
 - Retransmissions
 - Connection management
 - Flow control and buffer space
 - Congestion control
 - Principles
 - TCP congestion control
 - Evolving TCP and transport layer functionality



Thank you for joining today!

next:

- leaving the network “edge”
(application, transport layers)
- into the network “core”



Reading instructions chapter 3

- **KuroseRoss book**

Careful	Quick
3.1-3.7	

- **Other resources (further study)**

- Eddie Kohler, Mark Handley, and Sally Floyd. 2006. Designing DCCP: congestion control without reliability. *SIGCOMM Comput. Commun. Rev.* 36, 4 (August 2006), 27-38. <http://doi.acm.org/10.1145/1151659.1159918>
- Cardwell, Neal, et al. "BBR: congestion-based congestion control." *Communications of the ACM* 60.2 (2017): 58-66.
- Exercise/throughput analysis TCP in extra slides

Some more review questions

- Is TCP stateful or stateless?
- What is the trade-off to balance when setting TCP's timeout?
- Can an application have reliable data transfer if it uses UDP? If yes, how? If no, why?
- Why have TCP if we can add reliability on top of UDP inside an application protocol?
- Why does TCP use handshaking in the start and the end of connection?
- Why does TCP do fast retransmit upon a 3rd ack and not a 2nd?
- Describe TCP's flow control
- Describe TCP's congestion control: principle, method for detection of congestion, reaction.
- Can a TCP's session sending rate increase indefinitely?

Check also interactive questions/exercises at http://gaia.cs.umass.edu/kurose_ross/interactive/

Extra slides, for further study

From RFC 1122: TCP Ack

- TCP SHOULD implement a delayed ACK, but an ACK should not be excessively delayed; in particular, the delay MUST be less than 0.5 seconds, and in a stream of full-sized segments there SHOULD be an ACK for at least every second segment.
- A delayed ACK gives the application an opportunity to update the window and perhaps to send an immediate response. In particular, in the case of character-mode remote login, a delayed ACK can reduce the number of segments sent by the server by a factor of 3 (ACK, window update, and echo character all combined in one segment).
- In addition, on some large multi-user hosts, a delayed ACK can substantially reduce protocol processing overhead by reducing the total number of packets to be processed.
- However, excessive delays on ACK's can disturb the round-trip timing and packet "clocking" algorithms.
- We also emphasize that this is a SHOULD, meaning that an implementor should indeed only deviate from this requirement after careful consideration of the implications.

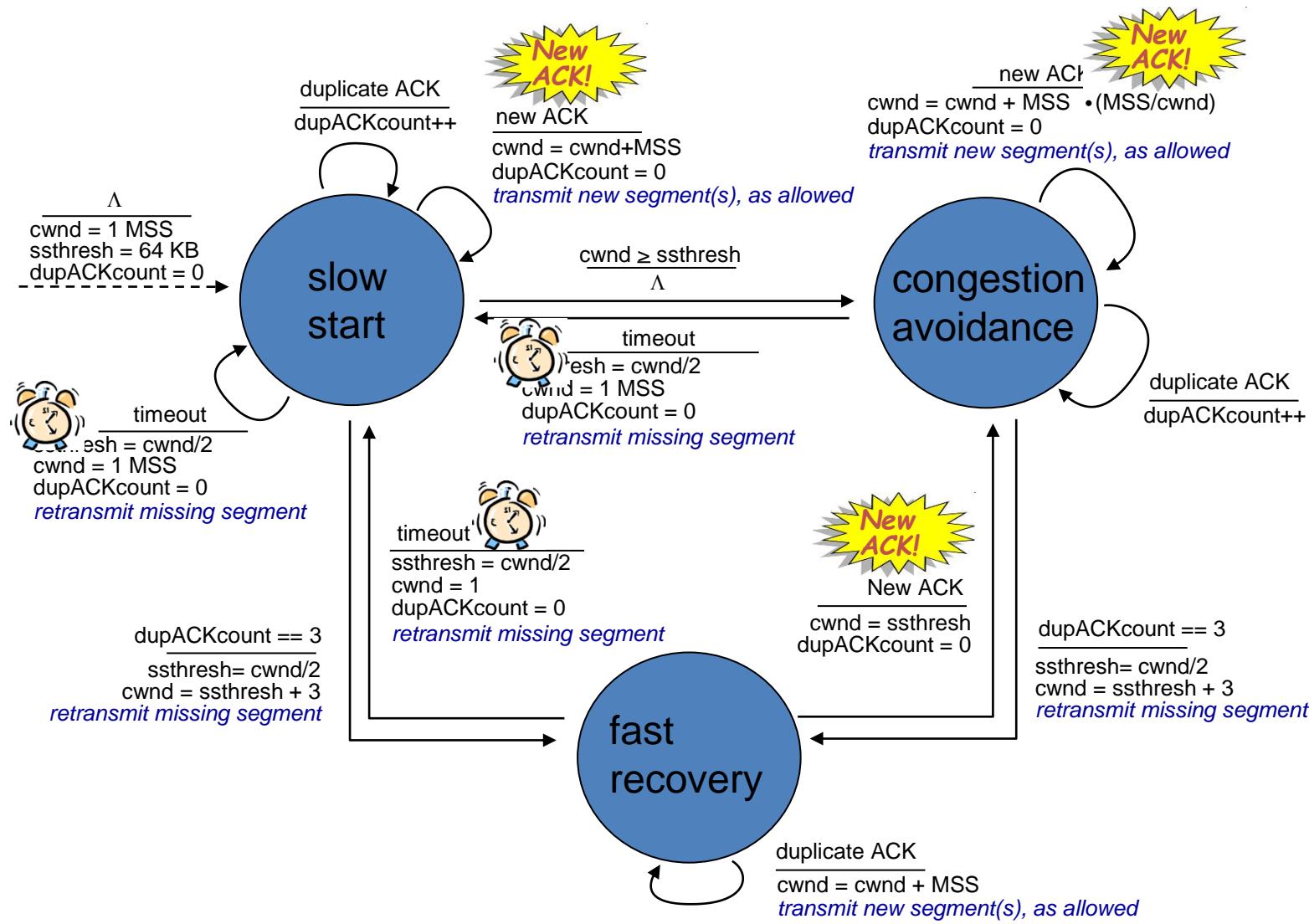
TCP – Closing a connection: Reset



RST

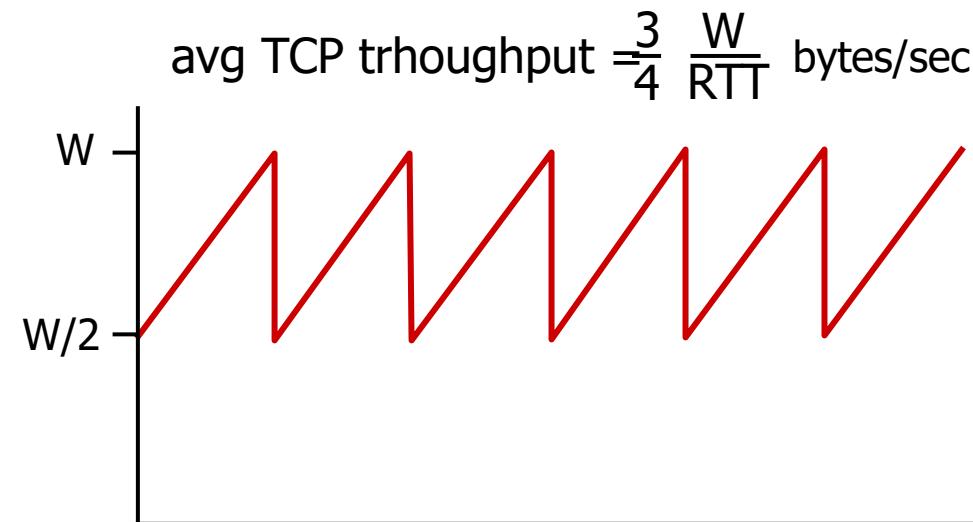
- RST is used to signal an error condition and causes an immediate close of the connection on both sides
- RST packets are not supposed to carry data payload, except for an optional human-readable description of what was the reason for dropping this connection.
- Examples:
 - A TCP data segment when no session exists
 - Connection attempt to non-existing port
 - Etc.

Summary: TCP Congestion Control



TCP throughput

- avg. TCP throughput as function of window size, RTT?
 - ignore slow start, assume always data to send
- W: window size (measured in bytes) where loss occurs
 - avg. window size (# in-flight bytes) is $\frac{3}{4} W$
 - avg. throughput is $\frac{3}{4}W$ per RTT



TCP Futures: TCP over “long, fat pipes”

- example: 1500 byte segments, 100ms RTT, want 10 Gbps throughput
- requires $W = 83,333$ in-flight segments
- throughput in terms of segment loss probability, L
[Mathis 1997]:
$$\text{TCP throughput} = \frac{1.22 \cdot \text{MSS}}{\text{RTT} \sqrt{L}}$$

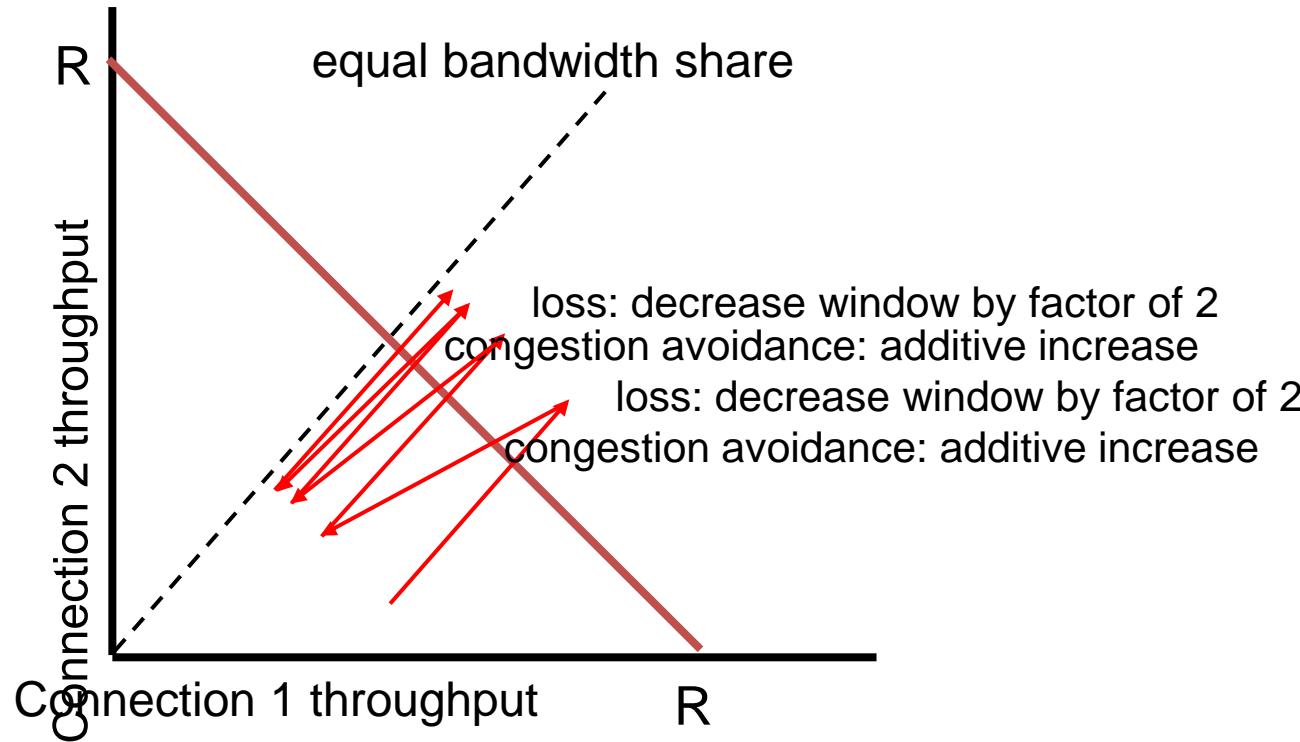
→ to achieve 10 Gbps throughput, need a loss rate of $L = 2 \cdot 10^{-10}$ – *a very small loss rate!*

- new versions of TCP for high-speed

Why is TCP fair?

two competing sessions:

- ❖ additive increase gives slope of 1, as throughout increases
- ❖ multiplicative decrease decreases throughput proportionally



Fairness, parallel TCP connections

- application can open multiple parallel connections between two hosts
- web browsers do this
- e.g., link of rate R with 9 existing connections:
 - new app asks for 1 TCP, gets rate $R/10$
 - new app asks for 11 TCPs, gets $R/2$

TCP delay modeling (slow start – related)

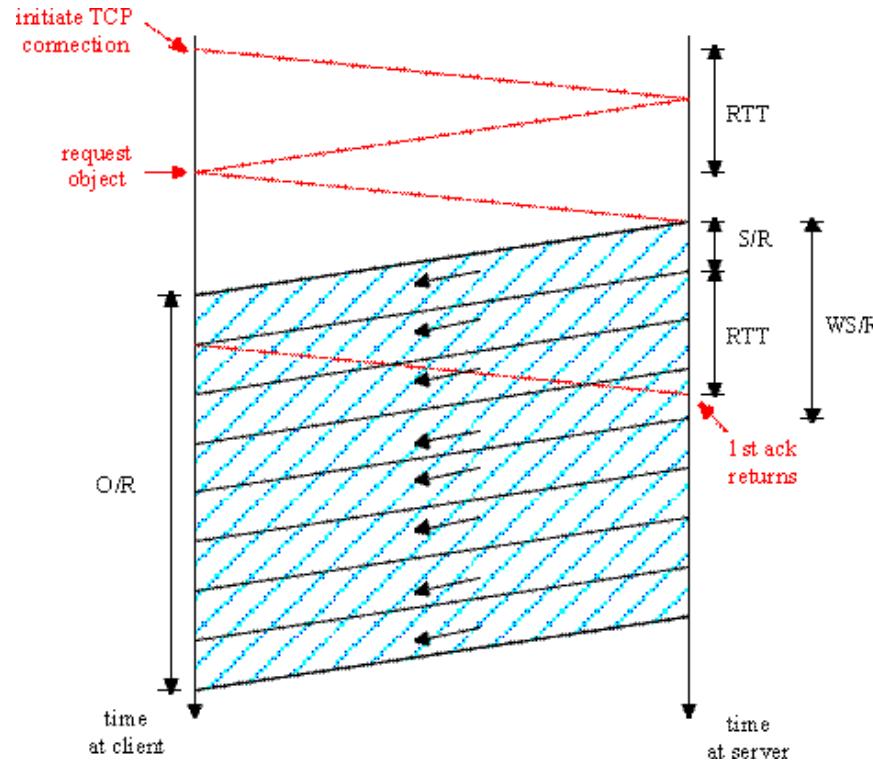
Q: How long does it take to receive an object from a Web server after sending a request?

- TCP connection establishment
- data transfer delay

Notation, assumptions:

- Assume one link between client and server of rate R
- Assume: fixed congestion window, W segments
- S : MSS (bits)
- O : object size (bits)
- no retransmissions (no loss, no corruption)
- Receiver has unbounded buffer

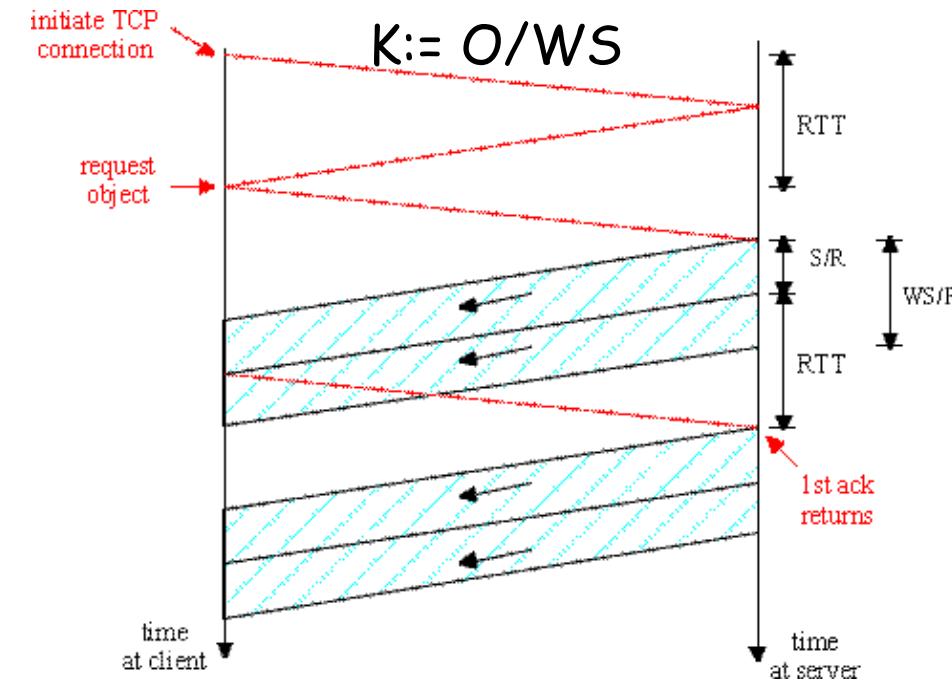
TCP delay Modeling: simplified, fixed window



Case 1 (cont. sendig): WS/R > RTT + S/R:

ACK for first segment in window returns before window's worth of data is sent

$$\text{delay} = 2\text{RTT} + \text{O/R}$$



Case 2 (pauses in sending) : WS/R < RTT + S/R:

wait for ACK after sending window's worth of data sent
 $\text{delay} = 2\text{RTT} + \text{O/R} + (\text{K}-1)[\text{S/R} + \text{RTT} - \text{WS/R}]$

TCP Delay Modeling: Slow Start

Delay components:

- 2 RTT for connection estab and request
- O/R to transmit object
- time server idles due to slow start

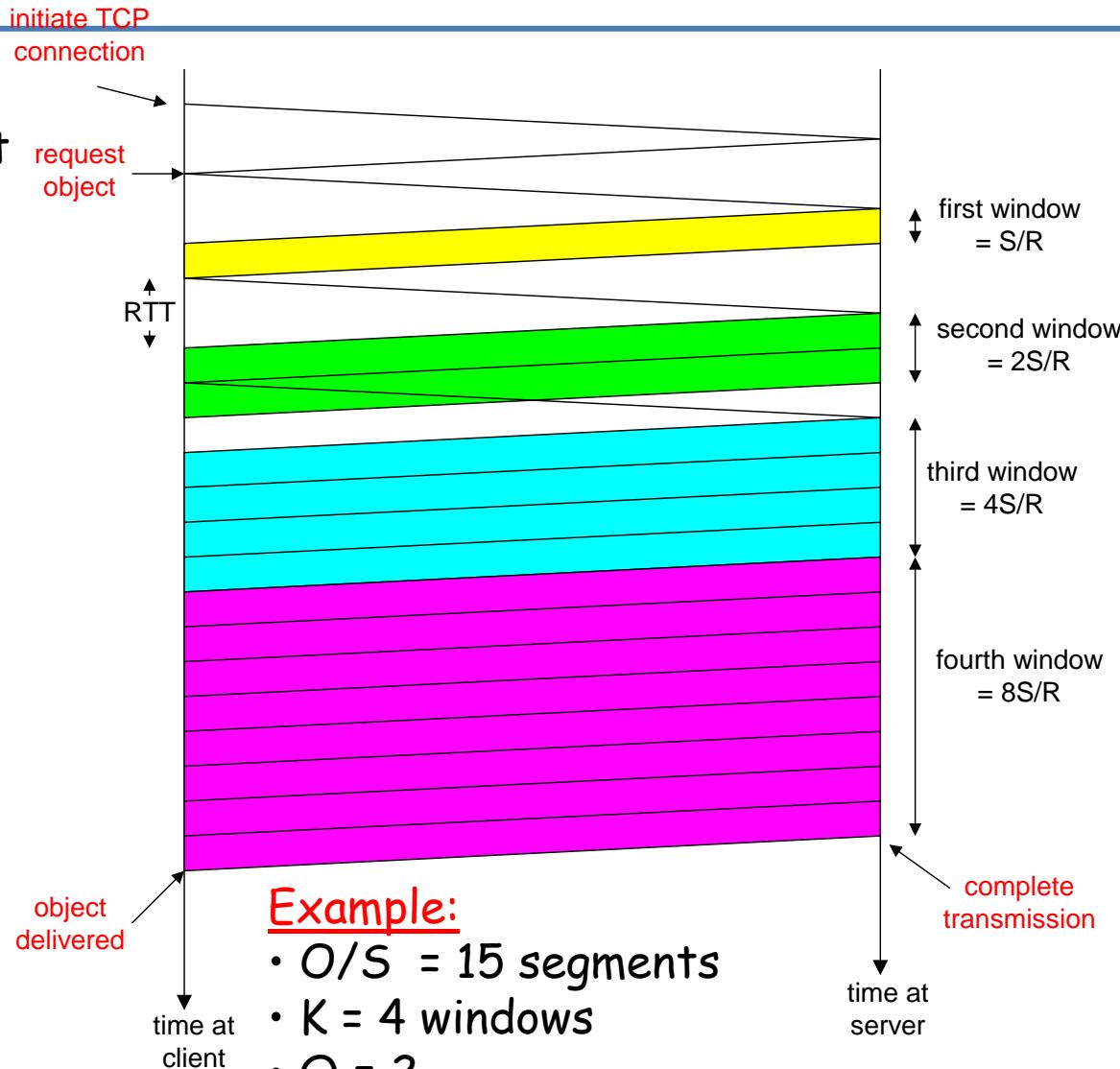
Server idles:

$$P = \min\{K-1, Q\} \text{ times}$$

where

- Q = #times server stalls until cong.
window is larger than a "full-utilization"
window (if the object were of unbounded
size).

- K = #(incremental-sized) congestion-
windows that "cover" the object.



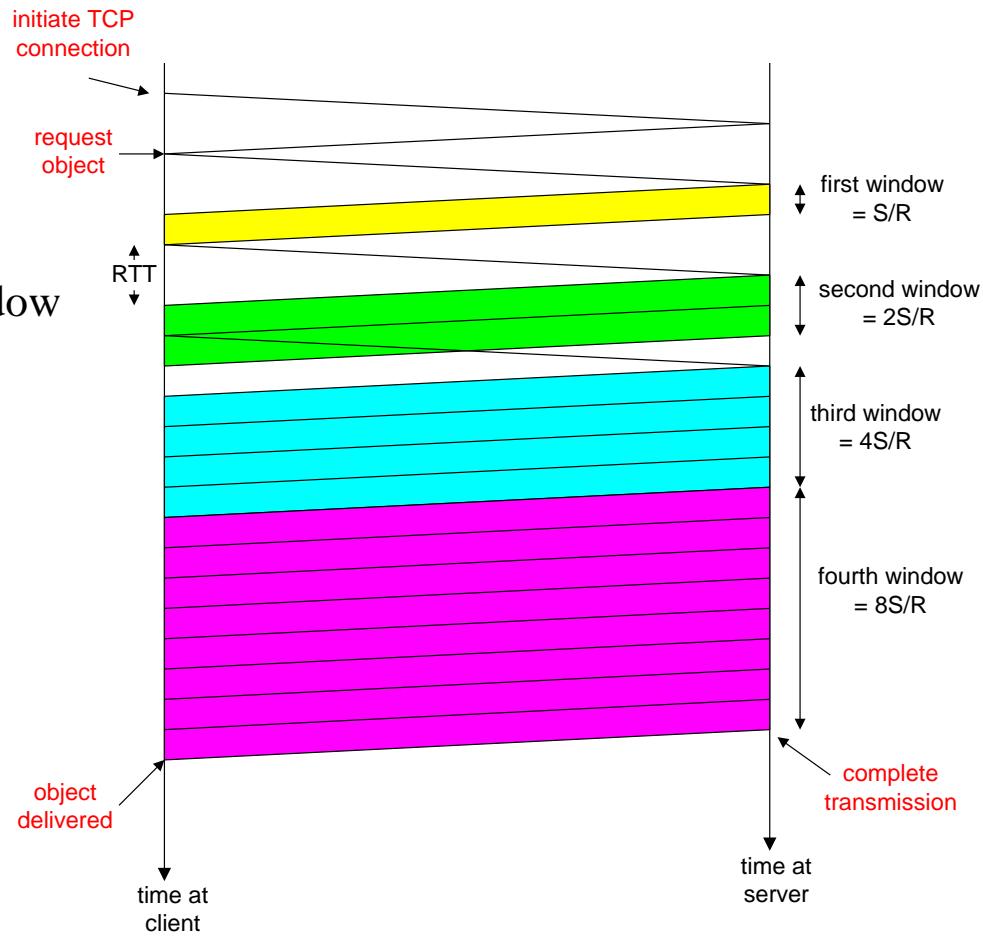
TCP Delay Modeling (slow start - cont)

$\frac{S}{R} + RTT =$ time from when server starts to send segment
until server receives acknowledgement

$2^{k-1} \frac{S}{R}$ = time to transmit the kth window

$\left[\frac{S}{R} + RTT - 2^{k-1} \frac{S}{R} \right]^+ =$ idle time after the kth window

$$\begin{aligned} \text{delay} &= \frac{O}{R} + 2RTT + \sum_{p=1}^P \text{idleTime}_p \\ &= \frac{O}{R} + 2RTT + \sum_{k=1}^P \left[\frac{S}{R} + RTT - 2^{k-1} \frac{S}{R} \right] \\ &= \frac{O}{R} + 2RTT + P[RTT + \frac{S}{R}] - (2^P - 1) \frac{S}{R} \end{aligned}$$



TCP Delay Modeling

Recall K = number of windows that cover object

How do we calculate K ?

$$\begin{aligned}K &= \min\{k : 2^0 S + 2^1 S + \dots + 2^{k-1} S \geq O\} \\&= \min\{k : 2^0 + 2^1 + \dots + 2^{k-1} \geq O/S\} \\&= \min\{k : 2^k - 1 \geq \frac{O}{S}\} \\&= \min\{k : k \geq \log_2(\frac{O}{S} + 1)\} \\&= \left\lceil \log_2(\frac{O}{S} + 1) \right\rceil\end{aligned}$$

Calculation of Q, number of idles for infinite-size object, is similar.



Course on Computer Communication and Networks

Lecture 6 Network Layer – Data Plane Chapter 4

EDA344 / DIT 423 / LEU062

Based on the book Computer Networking: A Top Down Approach, Jim Kurose, Keith Ross, Addison-Wesley.

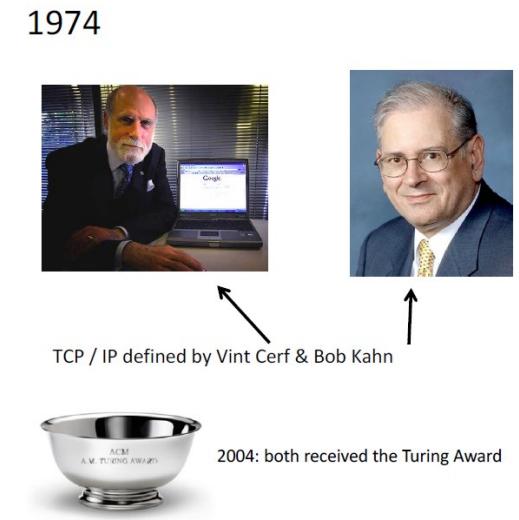
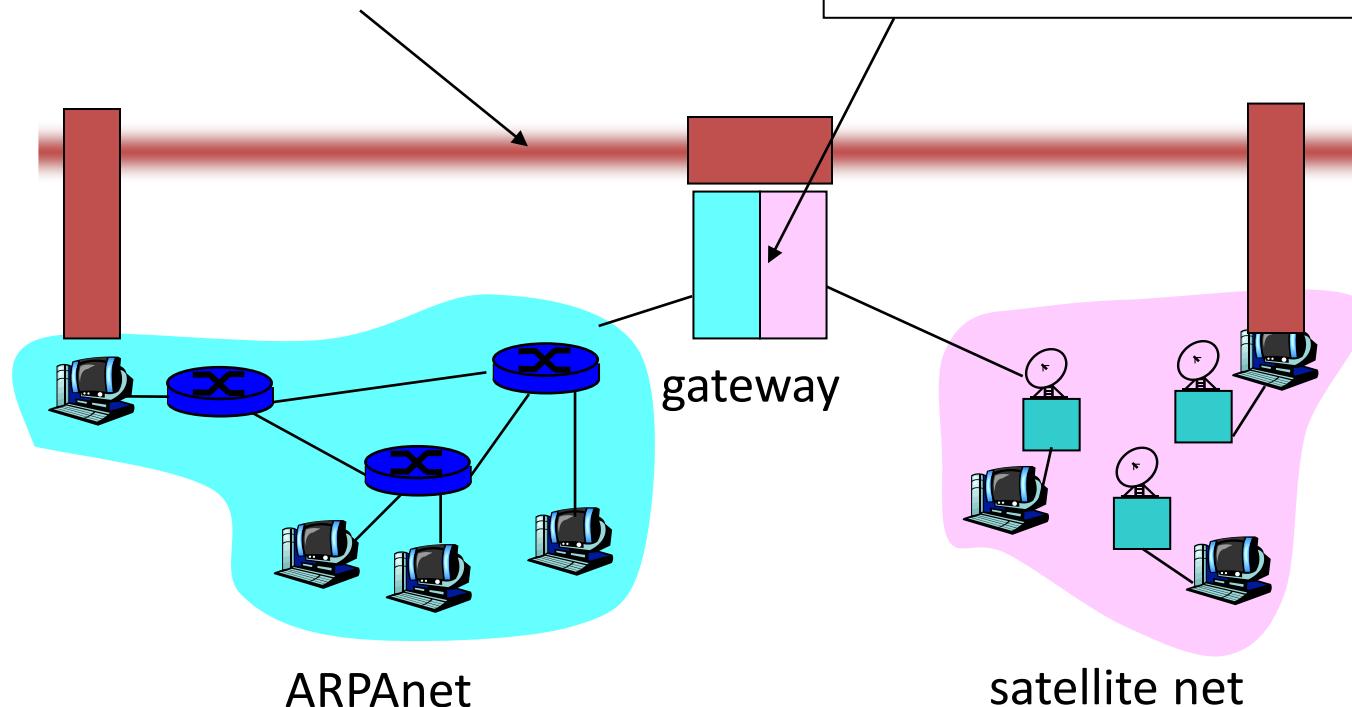
Recall: the Internet concept: bridging networks

Internet layer (IP):

- internetwork appears as single, uniform entity, despite underlying local network heterogeneity
- network of networks

Gateway:

- “embed internetwork packets in local packet format”
- route (at inter-network level) to next gateway

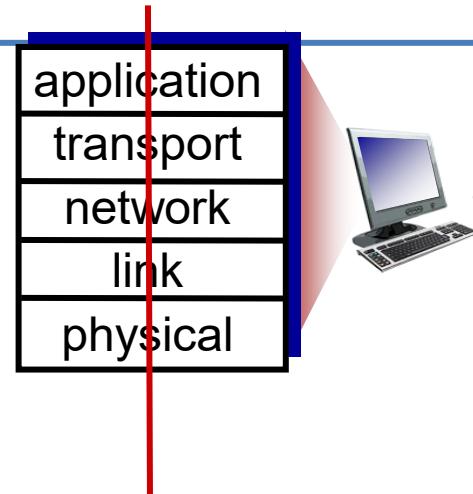


"A Protocol for Packet Network Intercommunication", V. Cerf, R. Kahn, IEEE Transactions on Communications, May, 1974, pp. 637-648.

Recall: network structure:

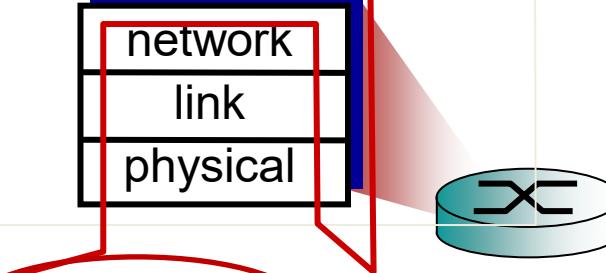
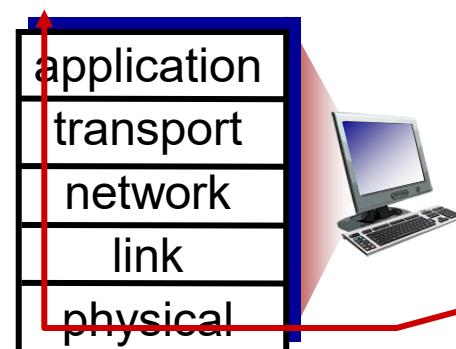
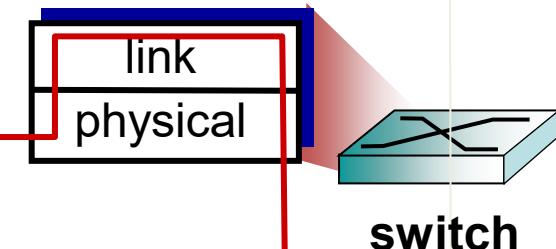
network edge: end-hosts:

- run application programs e.g. Web, email, ...
- ... based on network services available



access networks /Inter-connection:

- connect end-hosts to the Internet
- through physical media: wired, wireless links



network core:

- interconnected routers
- network of networks

Roadmap Network Layer

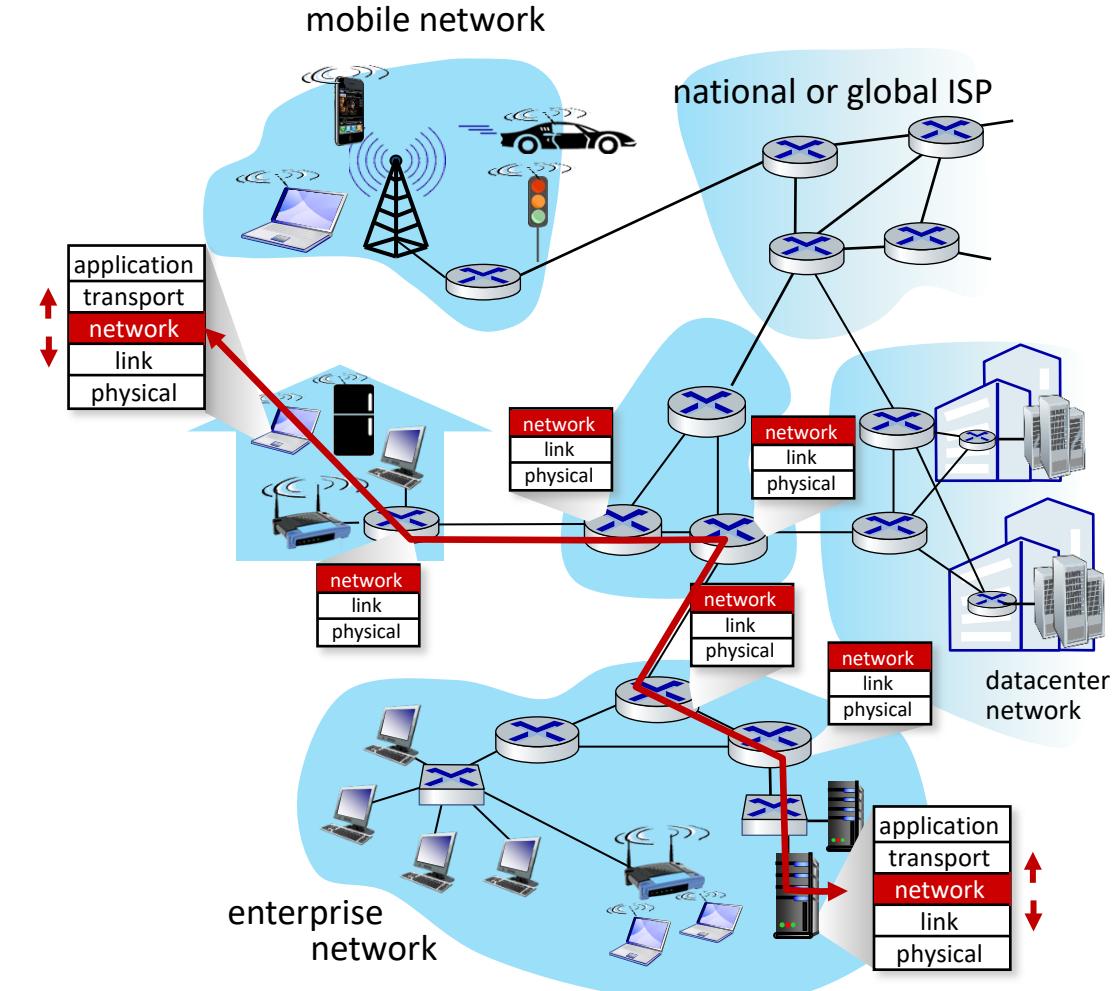


- Network layer functionality:
Forwarding vs routing
- Network layer service models
- Inside a router
- The Internet Network layer: IP,
Addressing & related



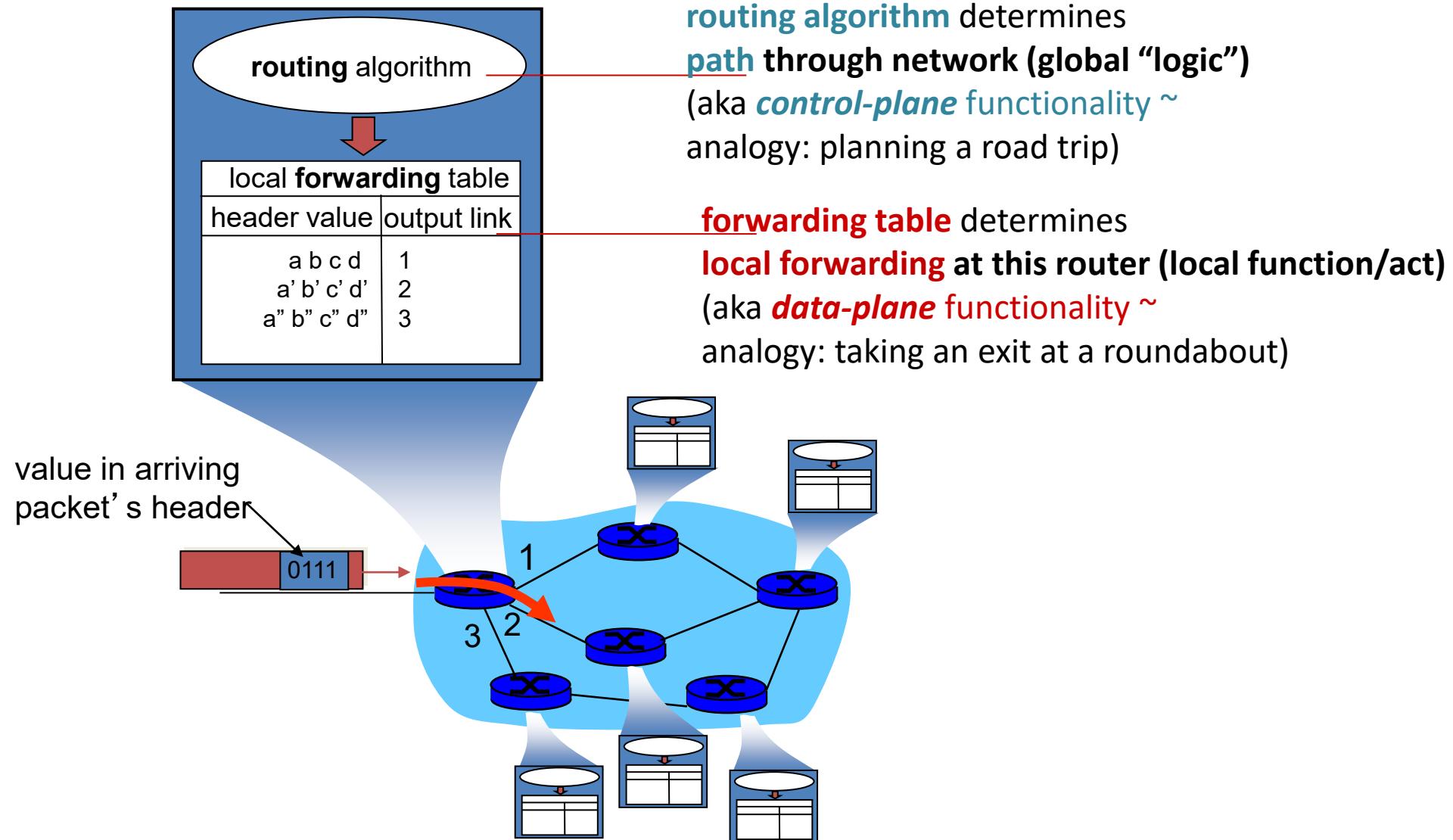
Network-layer functionality

- transport segment from sending to receiving host
 - sender: **encapsulates** segments into datagrams, passes to link layer
 - receiver: **delivers** segments to transport layer protocol
- network layer protocols in *every Internet device*: hosts, routers
- **router**:
 - examines header fields in all IP datagrams passing through
 - moves datagrams from input ports to output ports to transfer datagrams along end-end path



NW layer's job in more detail- routing and forwarding

Interplay between the two:



Synch:

1. In the discussed travel analogy, which of the following travel actions correspond to forwarding? (the others correspond to routing)

2. Which example actions are primarily in the network-layer data plane? (the other ones then correspond to control plane)

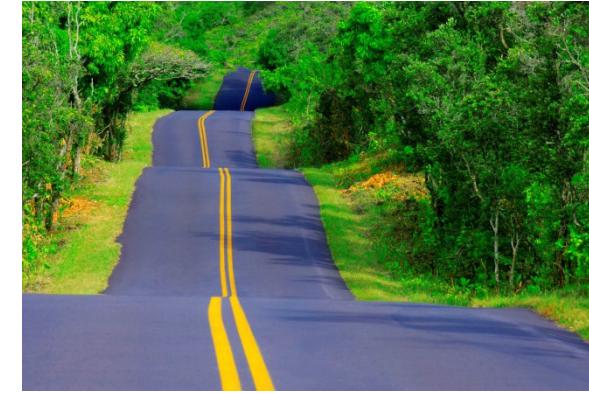
Go to
www.menti.com
Enter the code
8548 1231



Or use QR code

Roadmap Network Layer

- Forwarding versus routing
- Network layer service models
- Inside a router
- The Internet Network layer: IP, Addressing & related
- (Next) Control, routing
 - path selection
 - instantiation, implementation in the Internet



Network service model

Q: What *service model* for “channel” carrying packets from sender to receiver?

example services for individual packets:

- guaranteed delivery
- guaranteed delivery with less than 40 msec

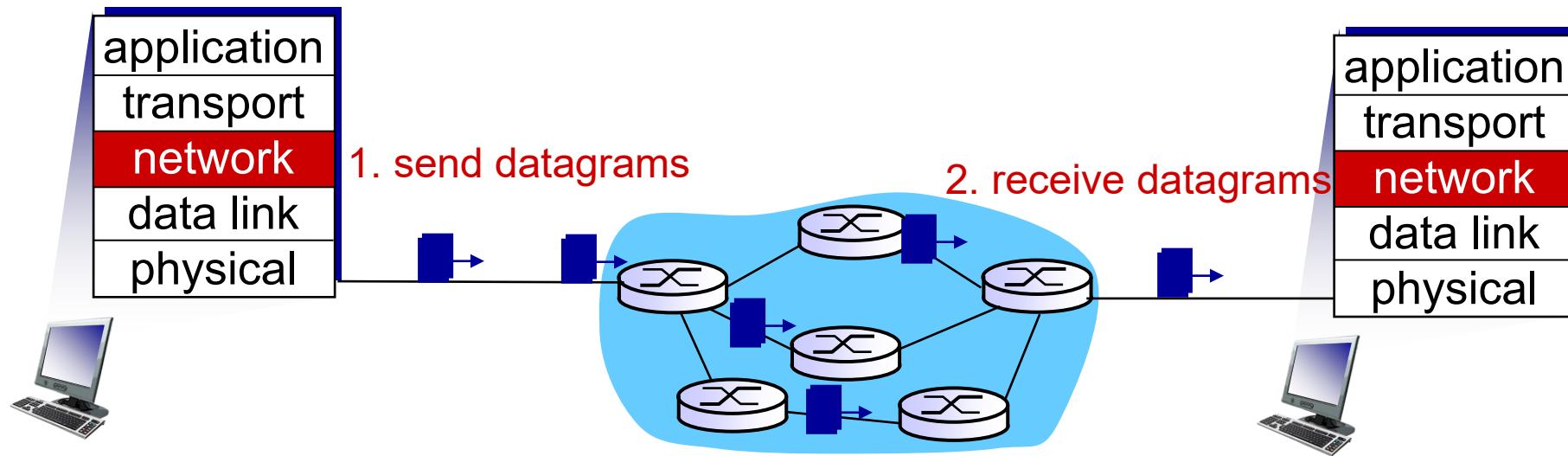
example services for a flow of packets:

- in-order delivery
- guaranteed min/avg/max arrival rate to flow

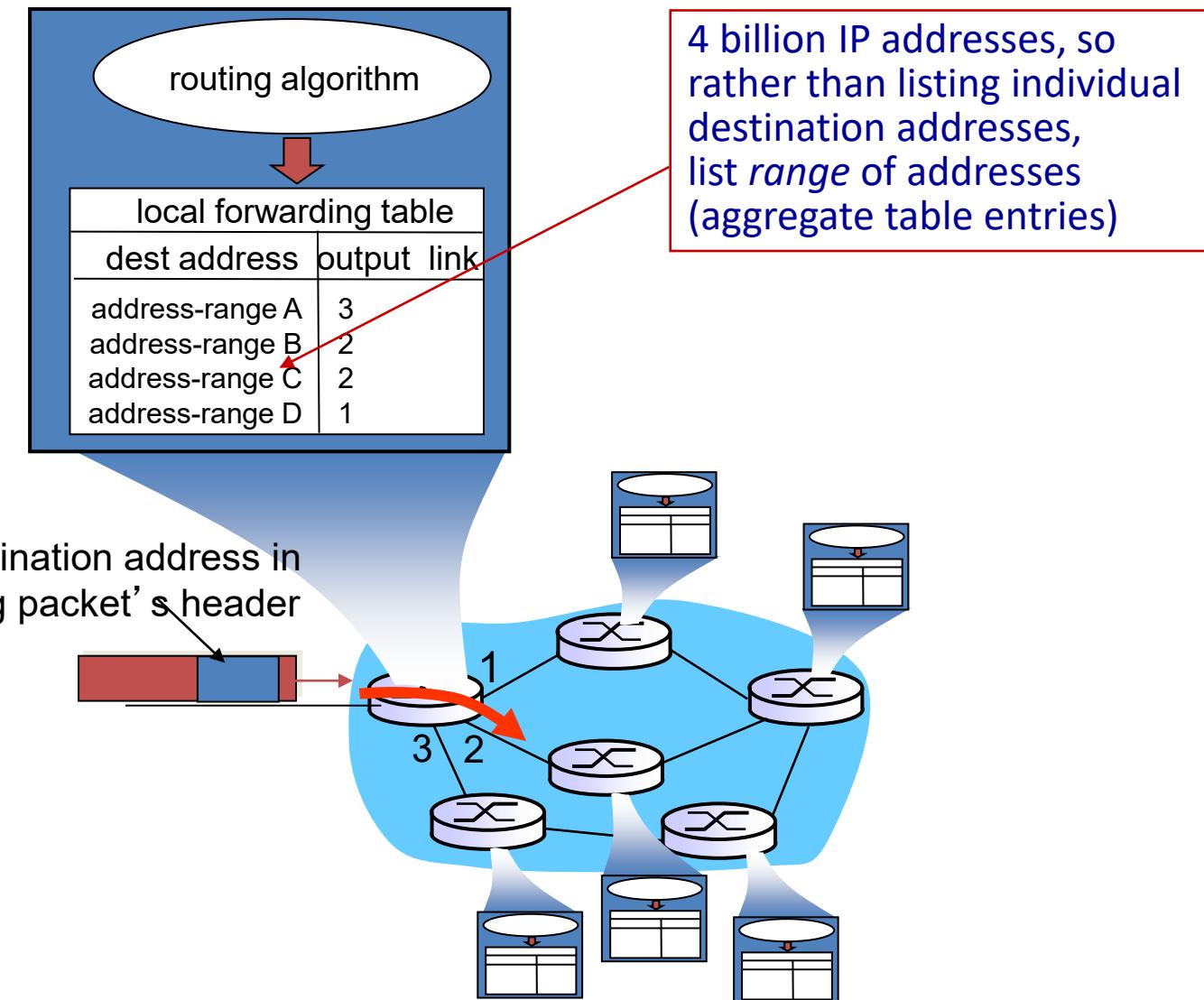
- *Internet's datagram* network provides network-layer *connectionless* service (i.e. none(!) of the above examples; **best effort delivery** instead)
 - classic Internet model

Datagram networks (classic Internet model)

- no call setup at network layer
- routers: **no state about end-to-end connections**
 - no network-layer concept of “connection”
- packets **forwarded using only destination host address**



Datagram (IP) forwarding table



An outlook on the basic NW-layer service models

Datagram, connectionless (Internet type)

- data exchange among **computers**
 - Best-effort service, no strict timing req.
- Why?
 - many link/network types, different characteristics => uniform service difficult
 - hourglass model
- “smart” end systems (computers)
 - can adapt, perform control, error recovery
 - **simple network core, complexity at edge**
 - (+ open-standard) allowed wide adoption

Virtual Circuit, connection-oriented:

- earlier vision of telecom for a WW-network(*)
- inspired from analog telephony, ~human conversation:
 - strict timing, reliability requirements
 - need for guaranteed service
- “dumb” end systems
 - **complexity in the core of network**
 - *Never fully implemented network stack*

(*)e.g. X-protocols and ATM (Asynchronous Transfer Mode) networks

CACM July 2019 - On The Hourglass Model - YouTube



Jun 26, 2019 - Uploaded by Association for Computing Machinery (ACM)
The **hourglass** model of layered systems architecture is a visual and conceptual representation of an approach ...

https://www.youtube.com/watch?v=L9s096_r_U

some adaptations in progress

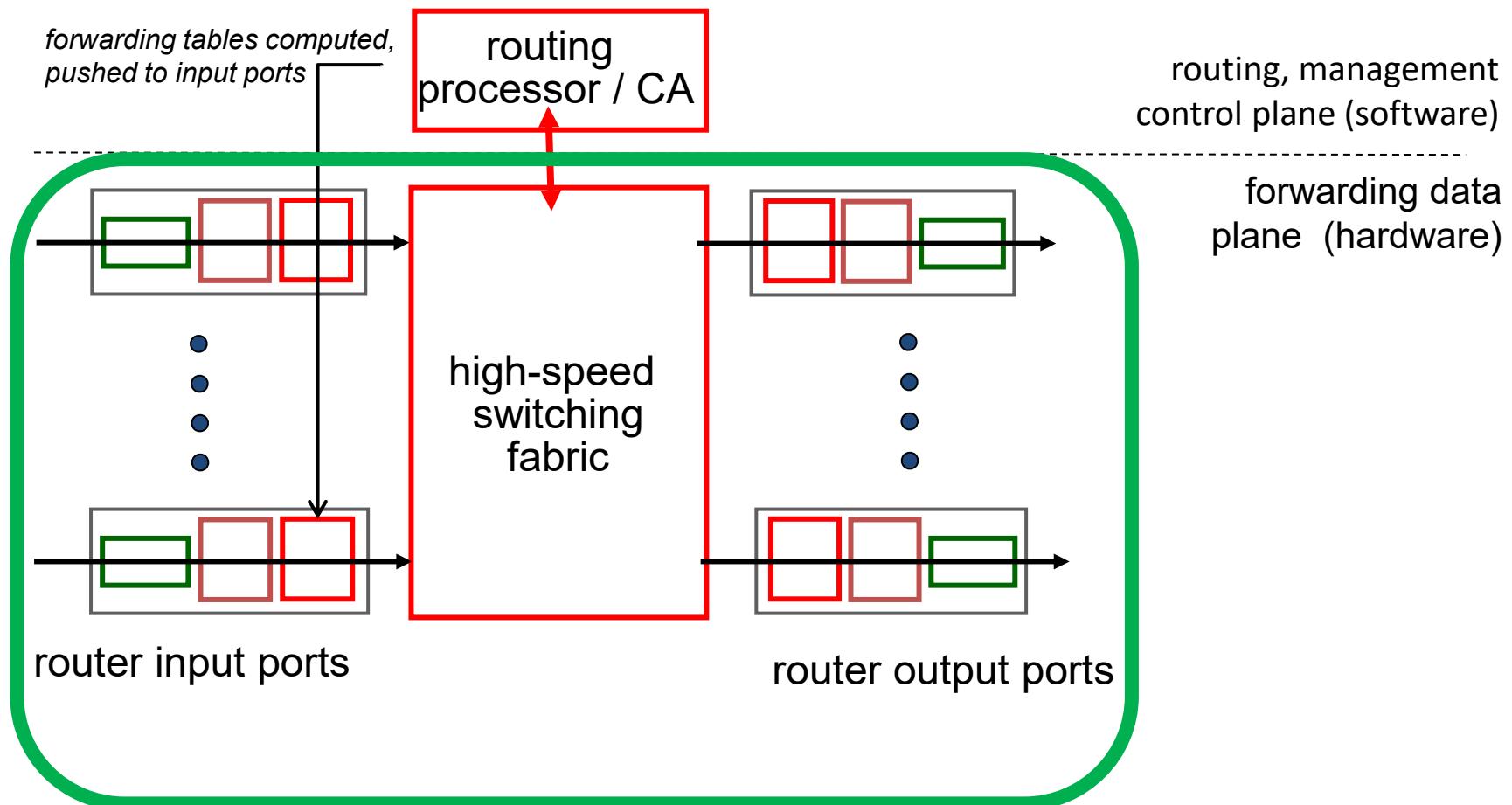
...with **Software-Defined Networks** (to be discussed later in the course)

Roadmap Network Layer

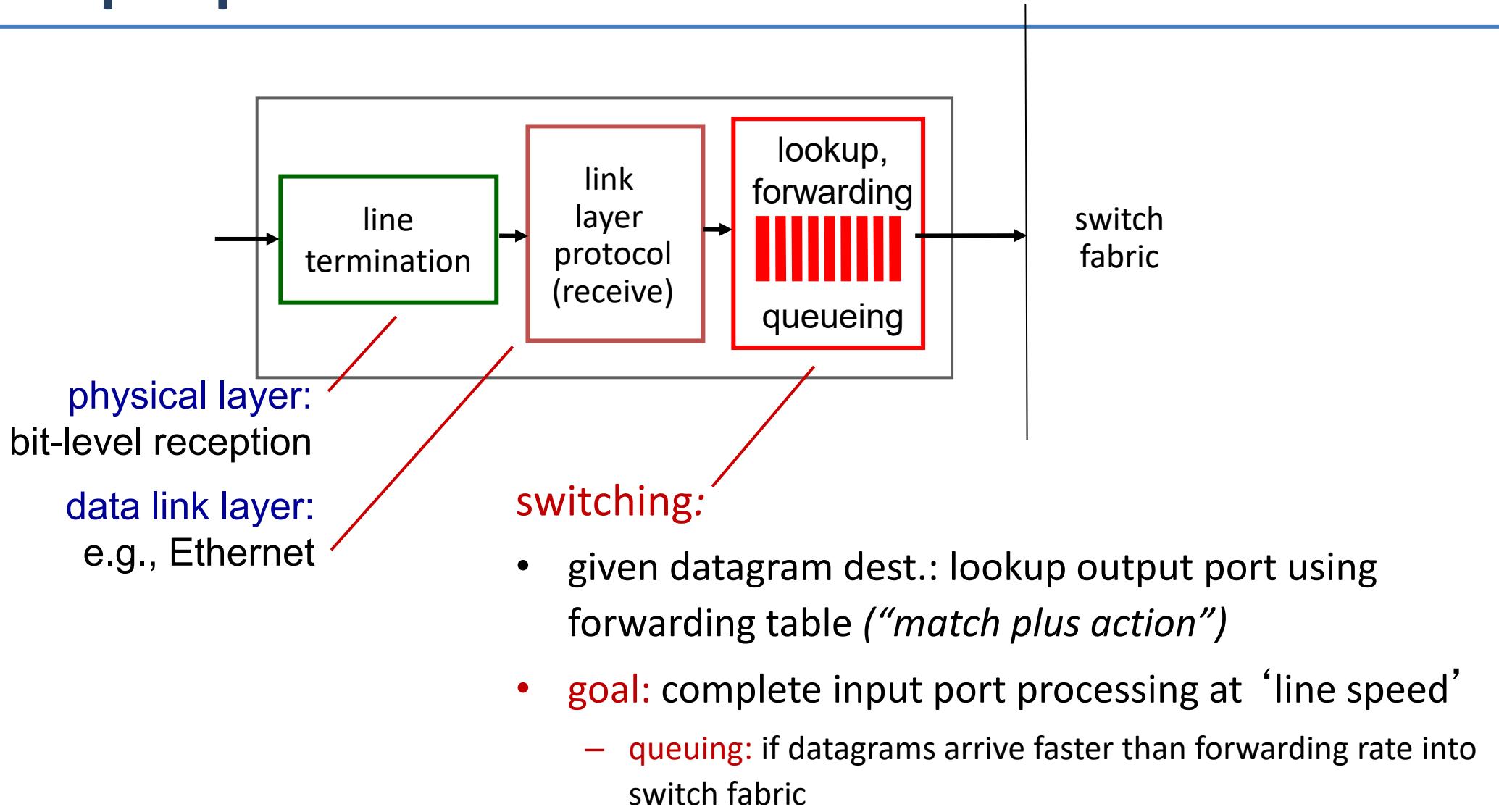
- Forwarding versus routing
- Network layer service models
- **Inside a router**
- The Internet Network layer: IP, Addressing & related
- (Next) Control, routing
 - path selection
 - instantiation, implementation in the Internet



Router architecture overview

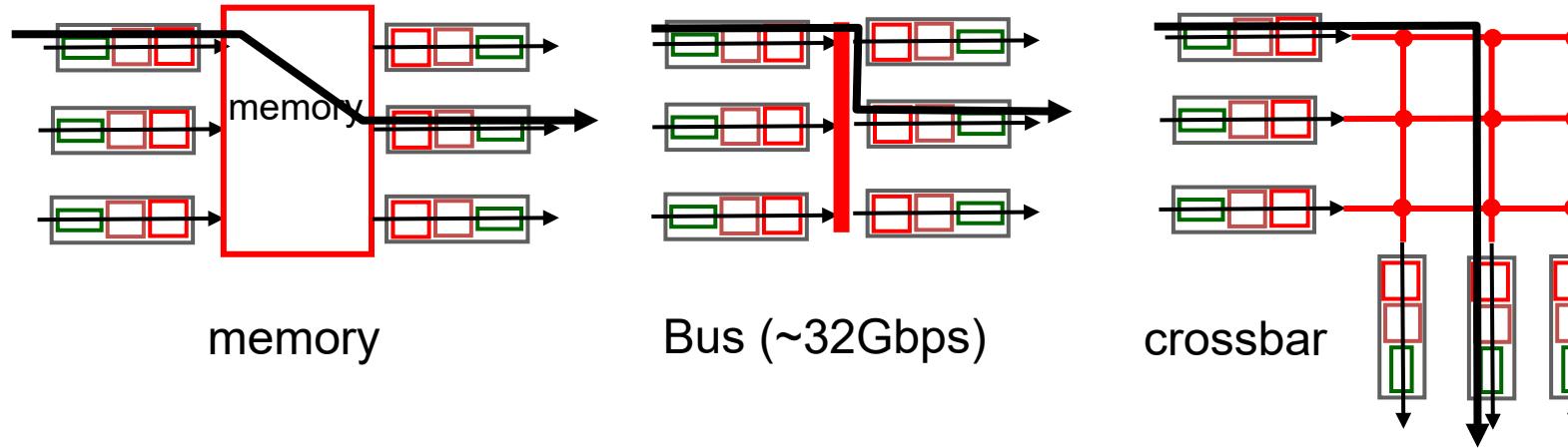


Input port functions



Switching fabrics

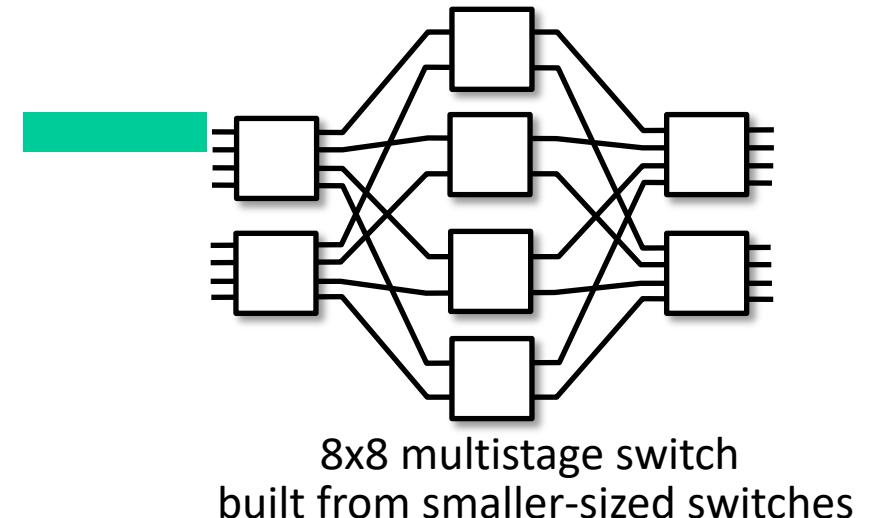
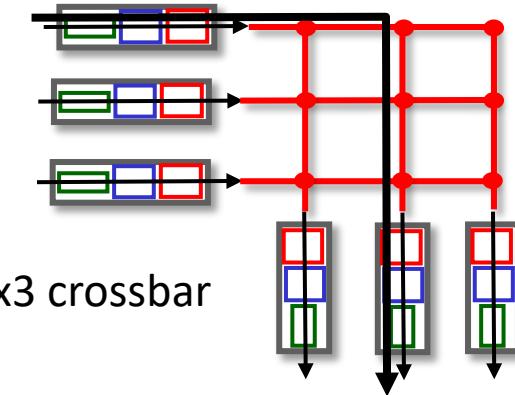
- switching rate: rate at which packets can be transferred from inputs to outputs
 - N links: desirable switching rate = $N * \text{link-rate}$
- three types:



Switching via interconnection network

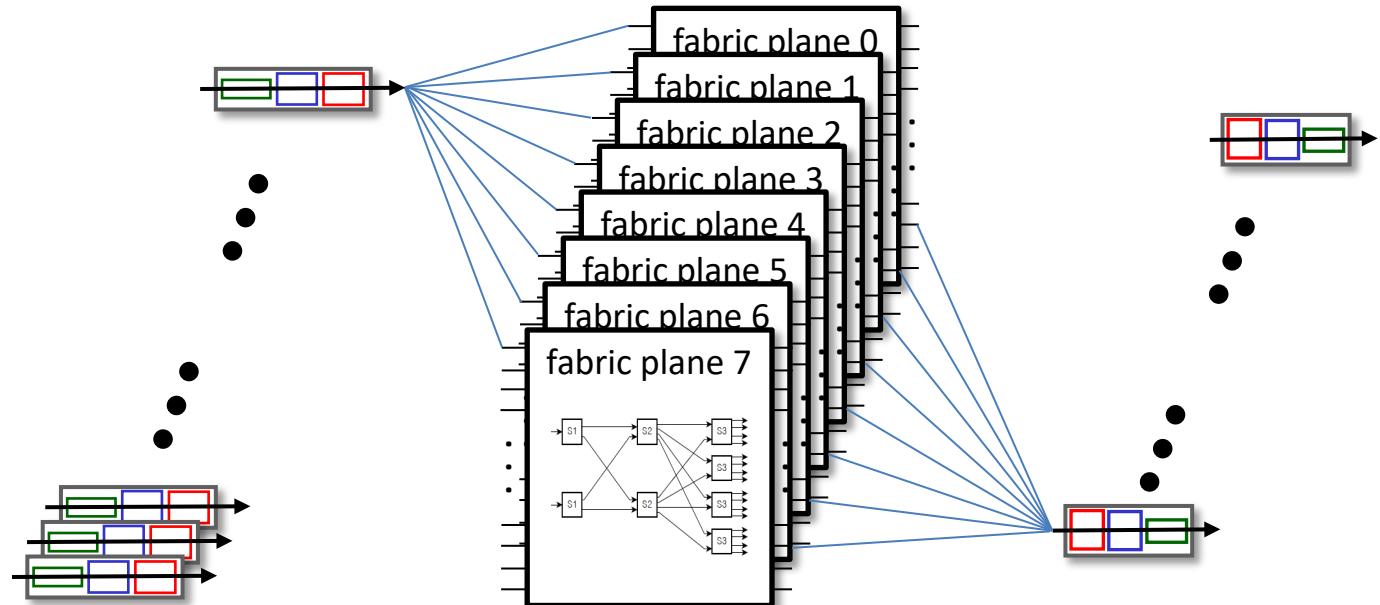
- initially developed to connect processors/memory in multiprocessors
- multistage switch: multiple stages of smaller switches
- exploiting parallelism:
 - fragment datagram into fixed length cells on entry^(*)
 - switch cells through the fabric, reassemble datagram at exit

(*) also ATM-associated technology



Switching via interconnection network

- scaling, using multiple switching “planes” in parallel:
 - speedup, scaleup via parallelism
- Cisco CRS router:
 - basic unit: 8 switching planes
 - each plane: 3-stage interconnection network
 - up to 100's Tbps switching capacity



Synch:

Where in a router is the destination IP address looked up in a forwarding table to determine the output port to direct the datagram ?

Go to
www.menti.com

Enter the code

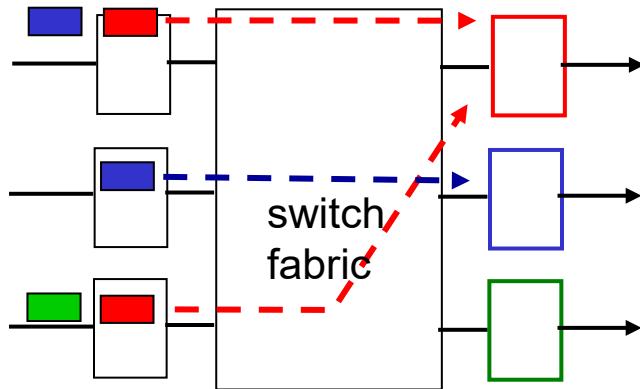
8548 1231



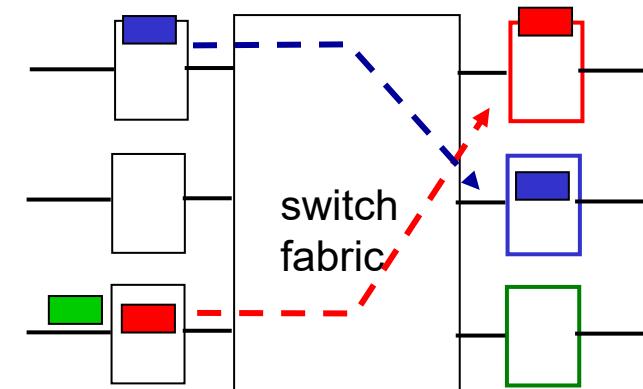
Or use QR code

Input port queuing

- If switch fabric slower than input ports combined -> queueing may occur at input queues
 - queueing delay and loss due to input buffer overflow!
- **Head-of-the-Line (HOL) blocking:** queued datagram at front of queue prevents others in queue from moving forward

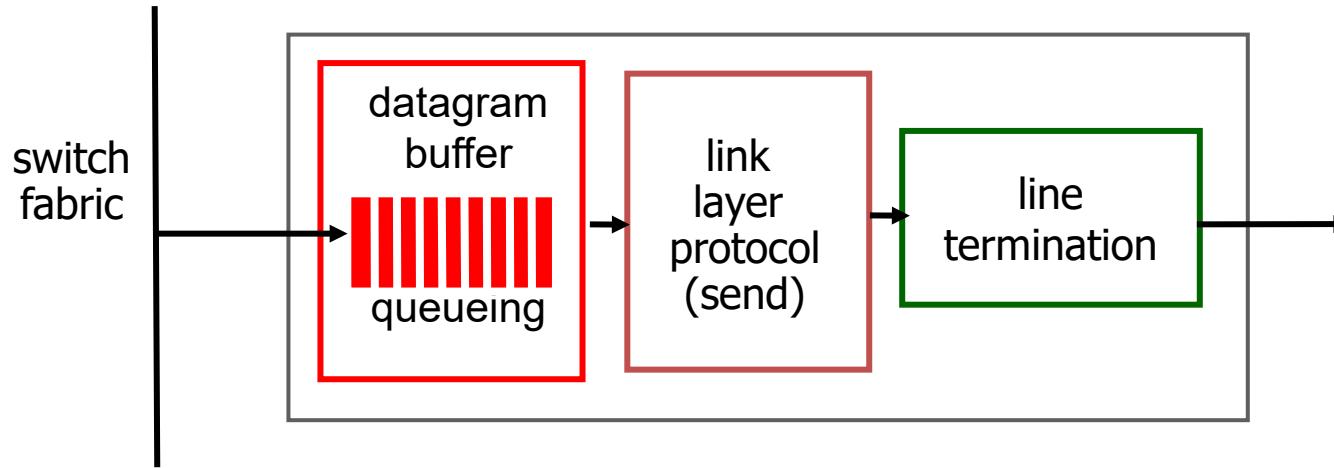


output port contention: only one red datagram can be transferred. lower red packet is *blocked*



one packet time later: green packet experiences HOL blocking

Output ports & queues



- *Buffering/queuing*: when datagrams arrive from fabric faster than the transmission rate
- *scheduling discipline* chooses among queued datagrams for transmission

Datagram (packets) can be lost due to congestion, lack of buffers

Priority scheduling – who gets best performance
(vs network neutrality)

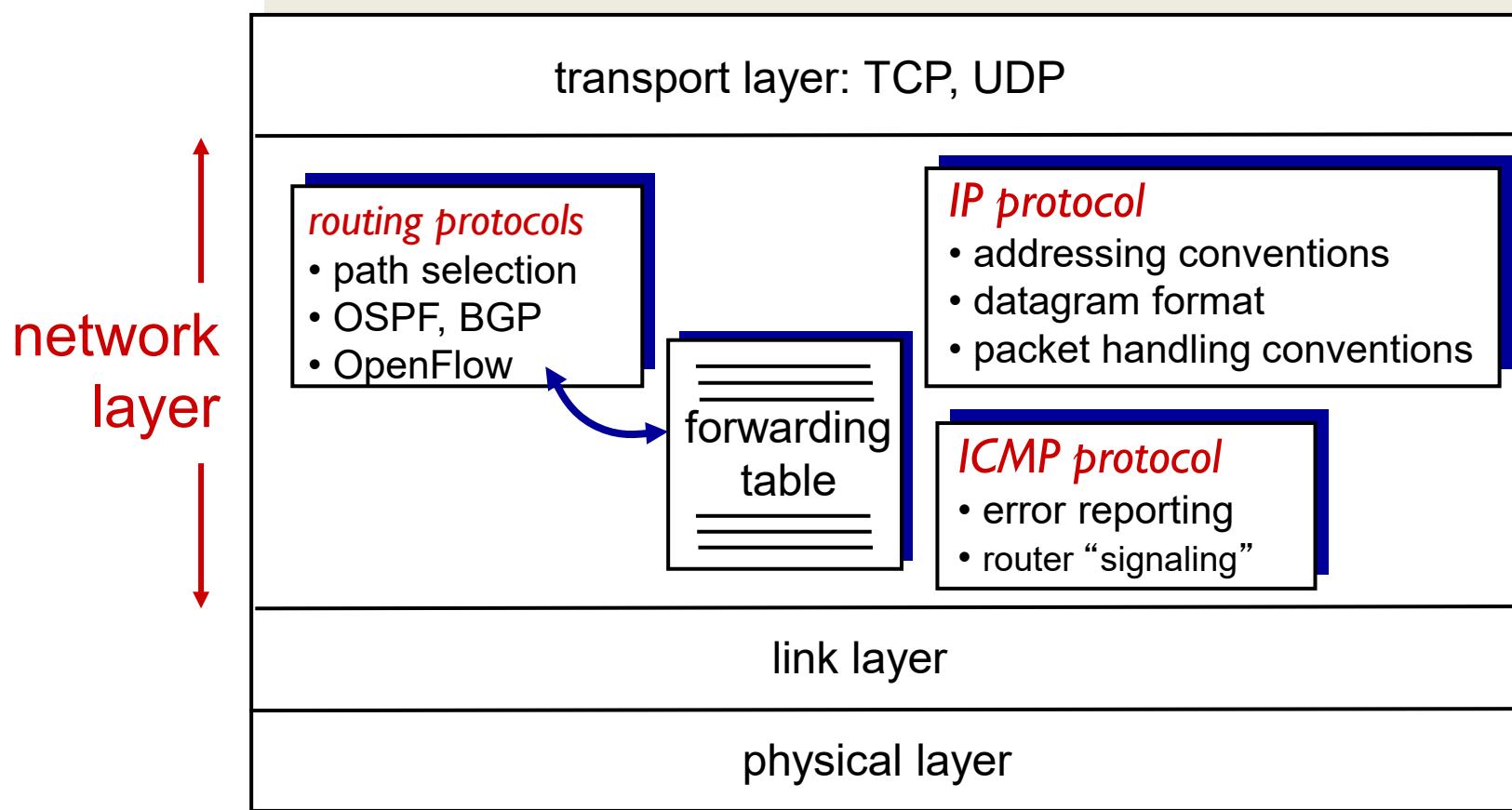
Roadmap Network Layer

- Forwarding versus routing
- Network layer service models
- Inside a router
- **The Internet Network layer: IP, Addressing & related**
- (Next) Control, routing
 - path selection
 - instantiation, implementation in the Internet

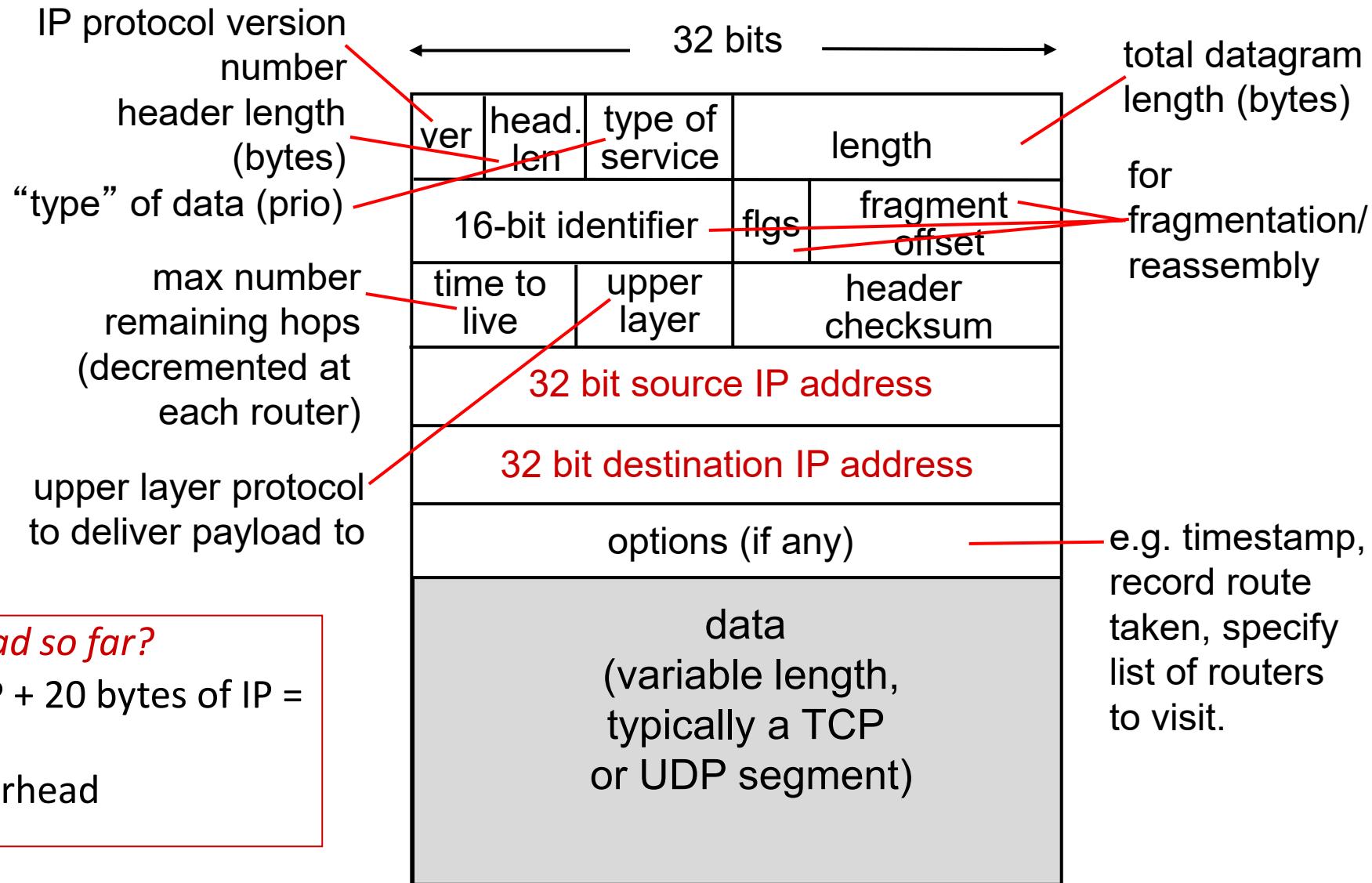


The Internet network layer

host, router network layer functions:



IPv4 datagram format



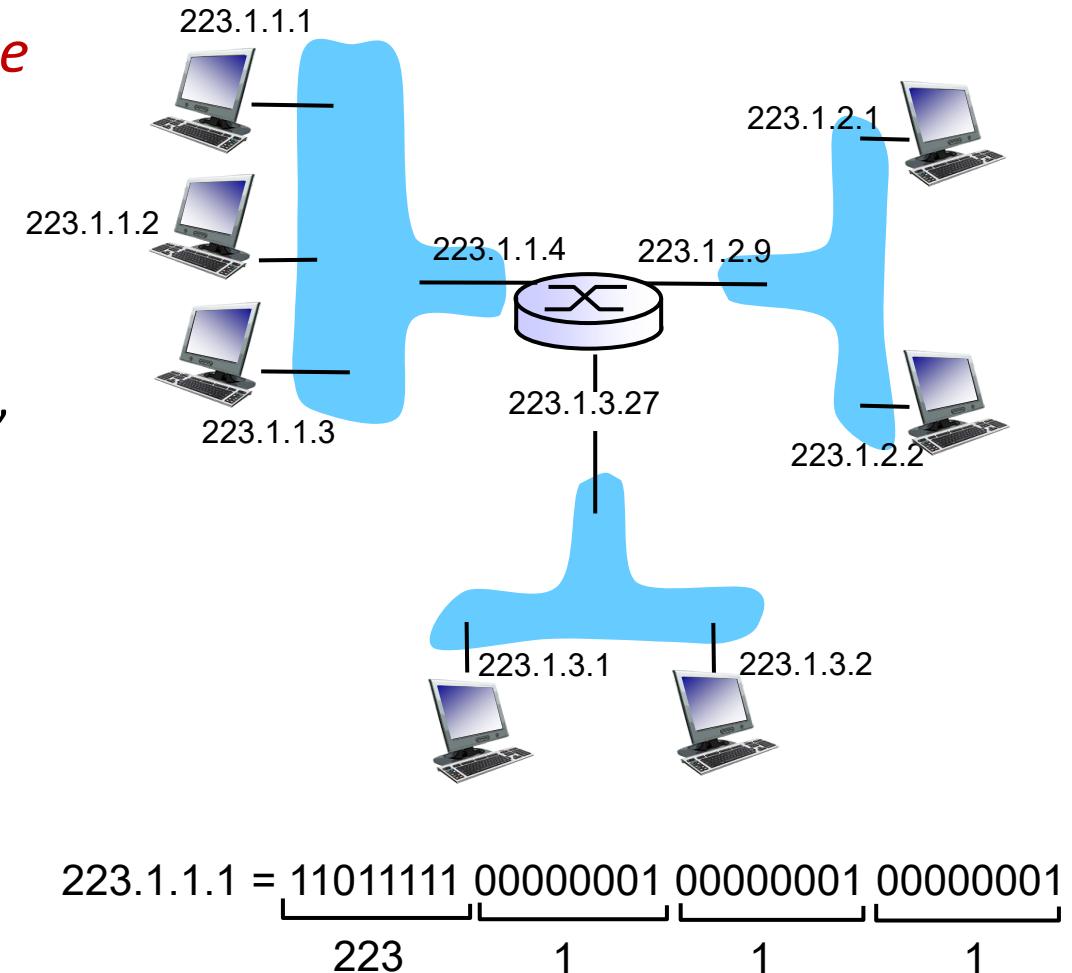
Roadmap Network Layer

- Forwarding versus routing
- Network layer service models
- Inside a router
- The Internet Network layer: IP, Addressing & related
 - Hierarchical addressing
- (Next) Control, routing
 - path selection
 - instantiation, implementation in the Internet



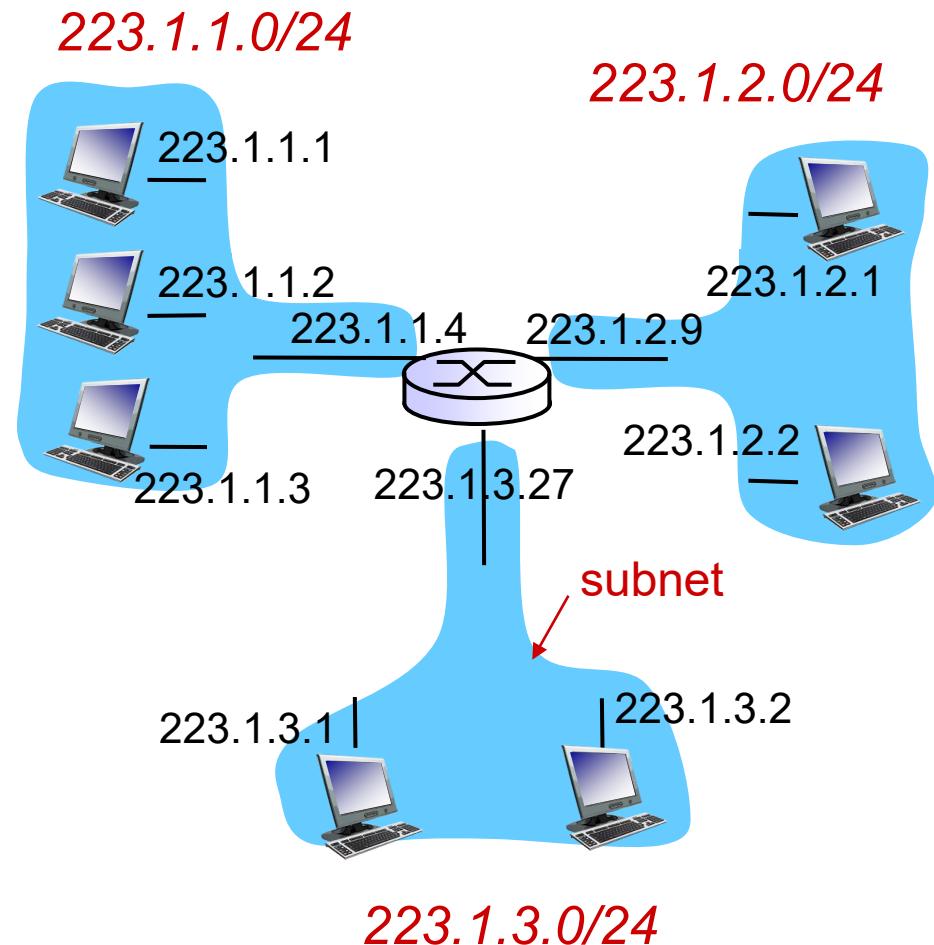
IP addressing: introduction

- **IP address:** 32-bit id for host/router *interface*
- **interface:** connection between host/router and physical link
 - routers have multiple interfaces
 - common end-host typically has 1-2 interfaces (e.g., wired Ethernet and wireless 802.11); servers can have more



Subnets

- IP address:
 - subnet part: high order bits (variable number)
 - host part: low order bits
- *what's a subnet ?*
 - Devices that can physically reach each other *without intervening router*
 - device interfaces have same subnet-part (prefix) of IP address



subnet mask: eg /24
defines how to find the subnet part of the address ...

Subnets, masks, calculations

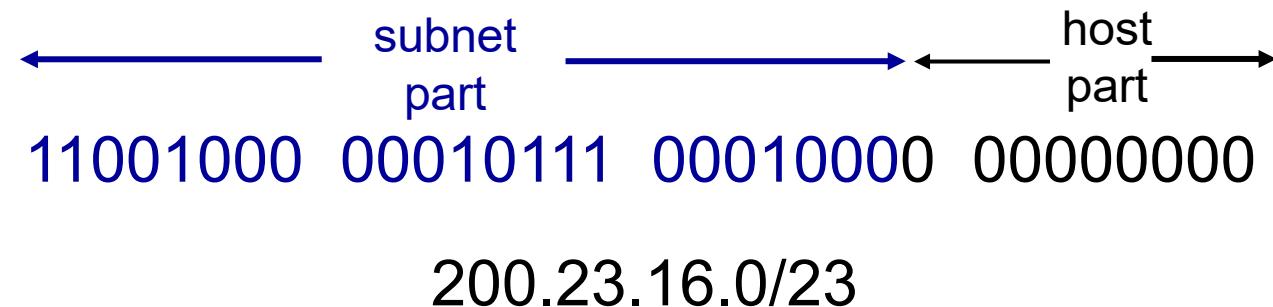
Example subnet: 192.168.5.0/24

	Binary form	Dot-decimal notation
IP address	11000000.10101000.00000101.10000010	192.168.5.130
Subnet mask	11111111.11111111.11111111.00000000 -----24 first bits set to 1-----	255.255.255.0
Network prefix: <i>bitwise AND of (address, mask)</i>	11000000.10101000.00000101.00000000	192.168.5.0
Host part (obtained with similar calculation, with a mask having the 8 last bits = 1)	00000000.00000000.00000000.10000010	0.0.0.130

IP addressing: CIDR

CIDR: Classless InterDomain Routing

- subnet portion of address of arbitrary length
- address format: **a.b.c.d/x**, where x is # bits in subnet portion of address



Roadmap Network Layer

- Forwarding versus routing
- Network layer service models
- Inside a router
- The Internet Network layer: IP, Addressing & related
 - Hierarchical addressing
 - Destination-based forwarding
- (Next) Control, routing
 - path selection
 - instantiation, implementation in the Internet



Destination-based forwarding

<i>forwarding table</i>	
Destination Address Range	Link Interface
11001000 00010111 00010000 00000000 through	0
11001000 00010111 00010000 00000100 through	3
11001000 00010111 00010000 00000111	
11001000 00010111 00011000 11111111	
11001000 00010111 00011001 00000000 through	2
11001000 00010111 00011111 11111111	
otherwise	3

Q: but what happens if ranges don't divide up so nicely?

Longest prefix matching

longest prefix match

when looking for forwarding table entry for given destination address, use *longest* address prefix that matches destination address.

Destination Address Range	Link interface
11001000 00010111 00010*** *****	0
11001000 00010111 00011000 *****	1
11001000 00010111 00011*** *****	2
otherwise	3

examples:

11001000 00010111 00010110 10100001 which interface?
11001000 00010111 00011000 10101010 which interface?

Longest prefix matching

longest prefix match

when looking for forwarding table entry for given destination address, use *longest* address prefix that matches destination address.

Destination Address Range	Link interface
11001000 00010111 00010*****	0
11001000 00010111 00011000 *****	1
11001000 1 00011*** *****	2
otherwise	3

examples:

11001000 00010111 00010110 10100001 which interface?

11001000 00010111 00011000 10101010 which interface?

Longest prefix matching

longest prefix match

when looking for forwarding table entry for given destination address, use *longest* address prefix that matches destination address.

Destination Address Range					Link interface
11001000	00010111	00010***	*****	*	0
11001000	00010111	00011000	*****	*	1
11001000	00010111	00011***	*****	*	2
otherwise					3

match!

examples:

11001000	00010111	00010110	10100001	which interface?
11001000	00010111	00011000	10101010	which interface?

Longest prefix matching

longest prefix match

when looking for forwarding table entry for given destination address, use *longest* address prefix that matches destination address.

Destination Address Range					Link interface
11001000	00010111	00010***	*****	*	0
11001000	00010111	00011000	*****	*	1
11001000	00010111	00011***	*****	*	2
otherwise					3

match!

examples:

11001000	00010111	00010110	10100001	which interface?
11001000	00010111	00011000	10101010	which interface?

Longest prefix matching

often performed using *ternary content addressable memories* (TCAMs)

- present address to TCAM: retrieve address in one clock cycle, regardless of table size
- Cisco Catalyst: ~1M table entries in TCAM

Synch:

On IP subnets

Go to
www.menti.com
Enter the code

8548 1231



Or use QR code

Roadmap Network Layer

- Forwarding versus routing
- Network layer service models
- How a router works
- The Internet Network layer: IP, Addressing & related
 - Hierarchical addressing
 - Destination-based forwarding
 - How to get addresses
- (Next) Control, routing
 - path selection
 - instantiation, implementation in the Internet



IP addresses: how to get one (for an end-host)?

hard-coded by system admin in a file

- Windows: control-panel->network->configuration->tcp/ip->properties;
- UNIX: /etc/rc.config
- Could not scale...

DHCP: Dynamic Host Configuration Protocol:
dynamically get address

DHCP returns more than just allocated IP address:

- IP address of first-hop router for client
- IP address of DNS sever
- network mask (network versus host portion of address)

DHCP server: 223.1.2.5



DHCP discover

src : 0.0.0.0, 68
dest.: 255.255.255.255, 67
yiaddr: 0.0.0.0 (your IP addr)
transaction ID: 654

arriving client



DHCP offer

src: 223.1.2.5, 67
dest: 255.255.255.255, 68
yiaddr: 223.1.2.4
transaction ID: 654
lifetime: 3600 secs

DHCP request

src: 0.0.0.0, 68
dest:: 255.255.255.255, 67
yiaddr: 223.1.2.4
transaction ID: 655
lifetime: 3600 secs

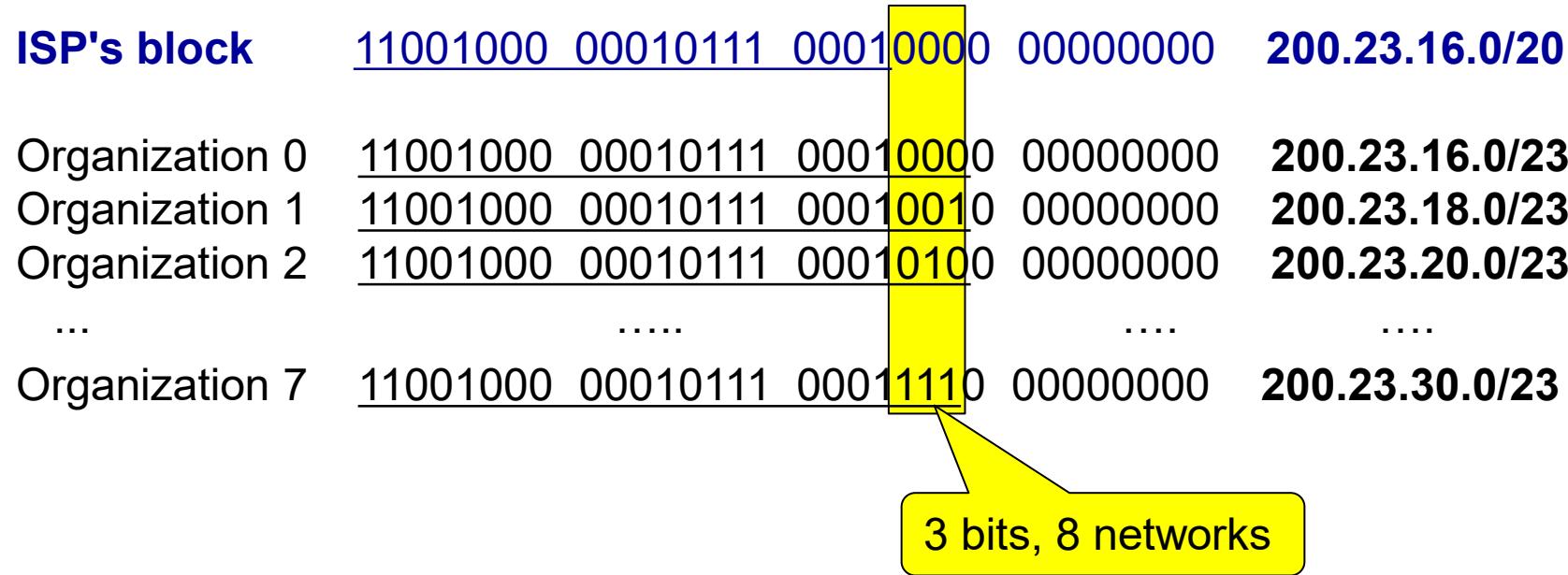
DHCP ACK

src: 223.1.2.5, 67
dest: 255.255.255.255, 68
yiaddr: 223.1.2.4
transaction ID: 655
lifetime: 3600 secs

IP addresses: how to get one (net-part)?

Q: how does *network* get subnet part of IP addr?

A: gets allocated portion of its provider ISP's address space; eg:



Classless Address Allocation: another example

- An ISP has an address block 122.211.0.0/16
- A customer needs max. 6 host addresses,
- ISP can e.g. allocate: 122.211.176.208/29
 - 3 bits enough for host part
- subnet mask 255.255.255.248

Reserved addresses

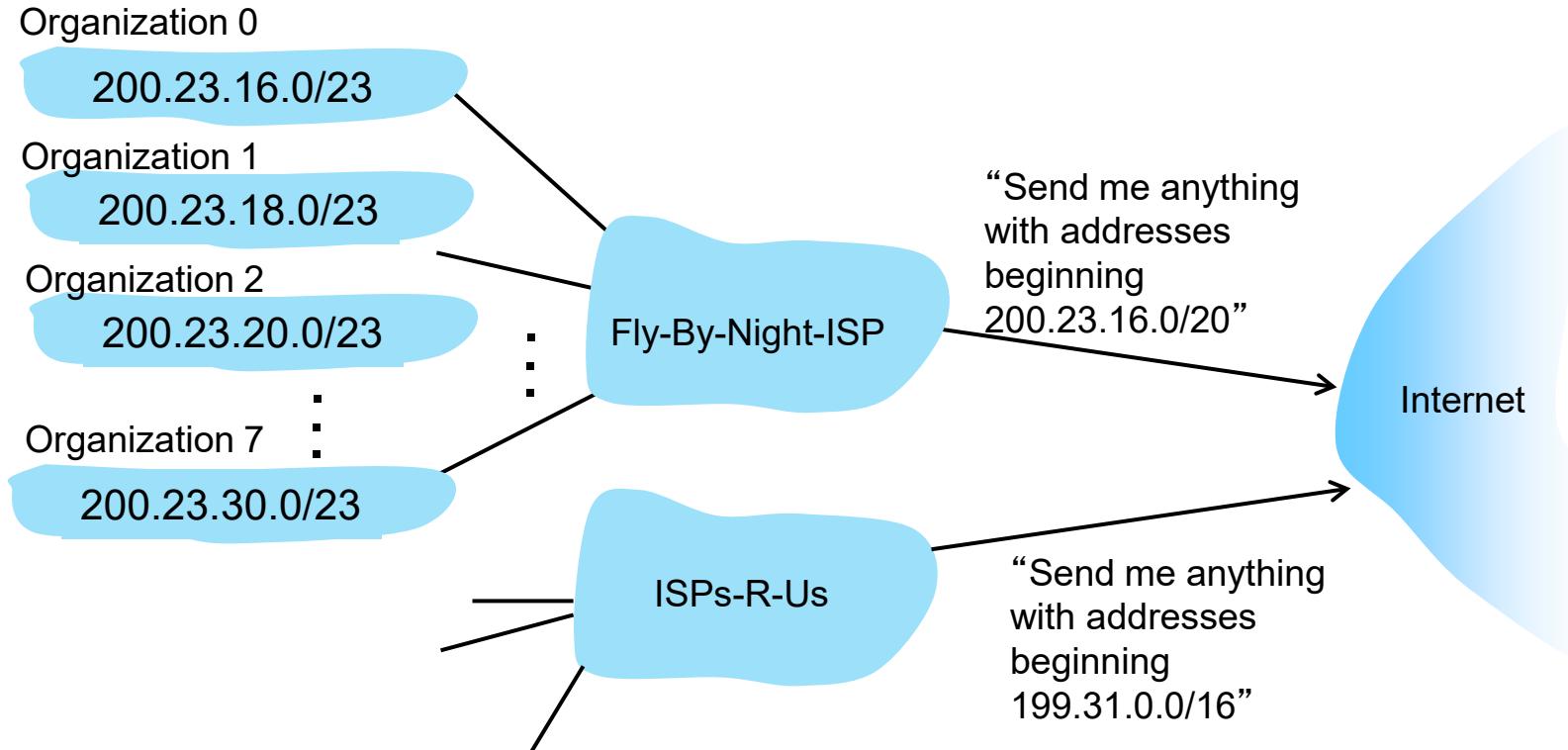
	Dotted Decimal	Last 8 bits
Network	122.211.176.208	11010000
1st address	122.211.176.209	11010001
.....
6th address	122.211.176.214	11010110
Broadcast	122.211.176.215	11010111

RFC 3021 “The network address itself {<Network-number>, 0} is an obsolete form of directed broadcast, but it may still be used by older hosts.”

Hierarchical addressing: route aggregation

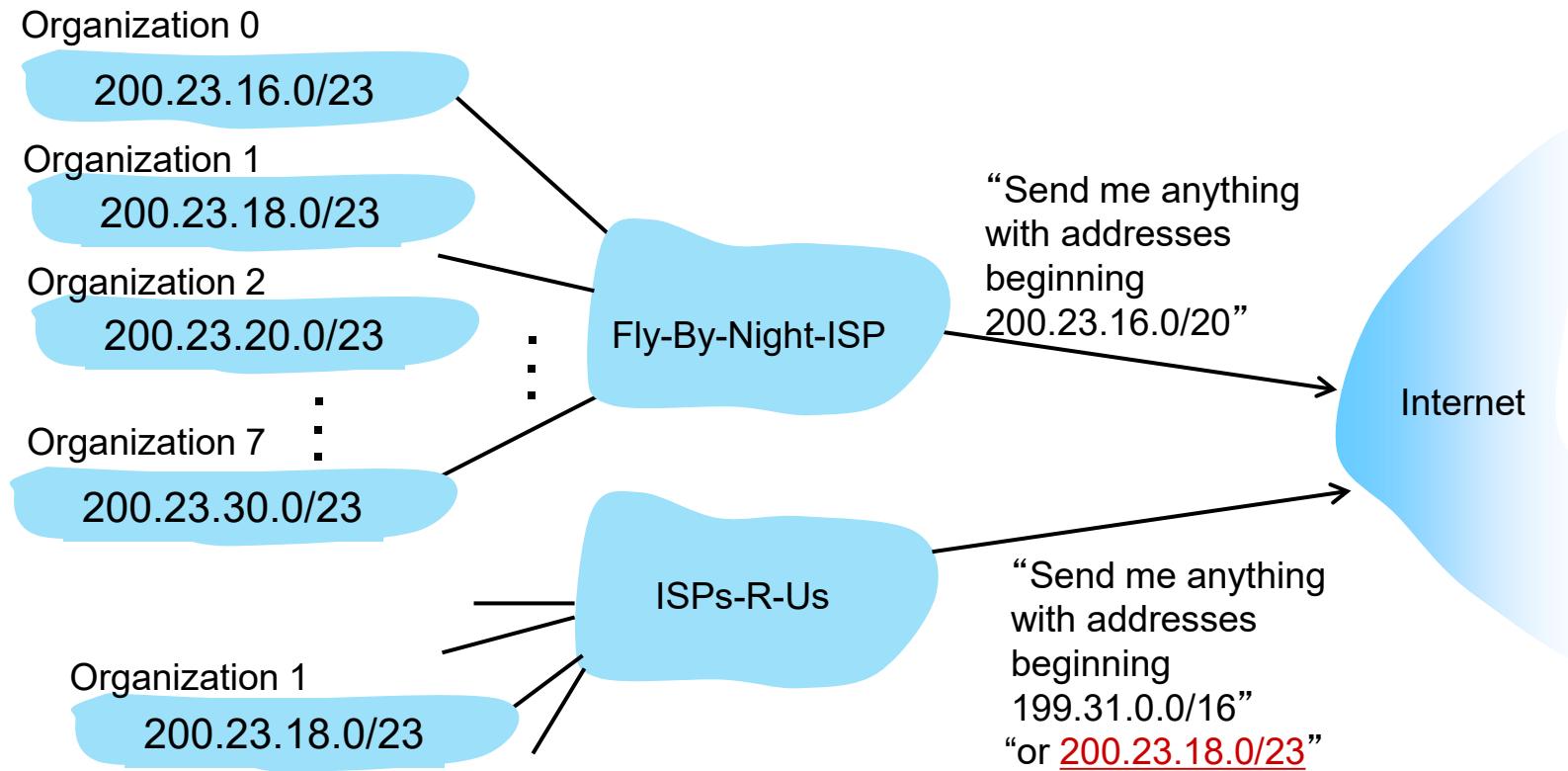
hierarchical addressing allows efficient *advertisement*^(*) of routing information:

()telling the other routers which addresses are to be found in this direction*



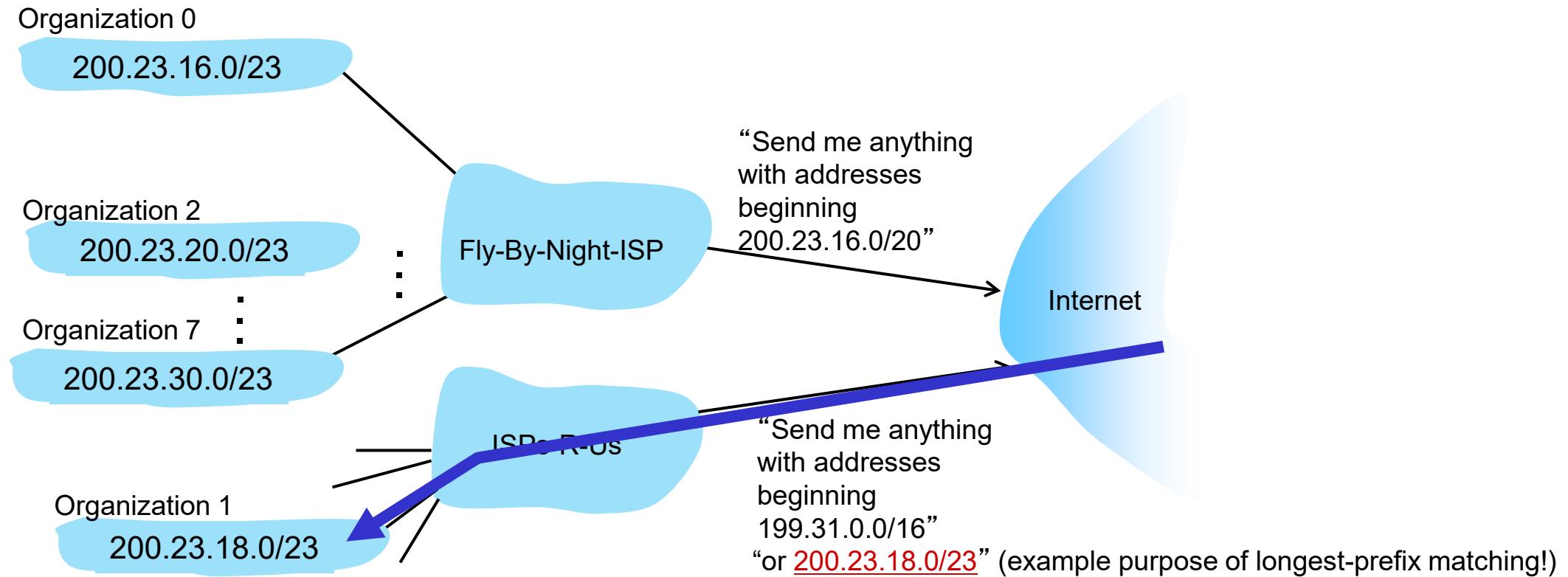
Hierarchical addressing: more specific routes

- Organization 1 moves from Fly-By-Night-ISP to ISPs-R-Us
- ISPs-R-Us now advertises a more specific route to Organization 1



Hierarchical addressing: more specific routes

- Organization 1 moves from Fly-By-Night-ISP to ISPs-R-Us
- ISPs-R-Us now advertises a more specific route to Organization 1



IP Addressing: the last word...

Q: How does an ISP get block of addresses?

A: ICANN: <http://www.icann.org/>

Internet Corporation for Assigned Names and Numbers



- allocates addresses
- manages DNS
- assigns domain names, resolves disputes

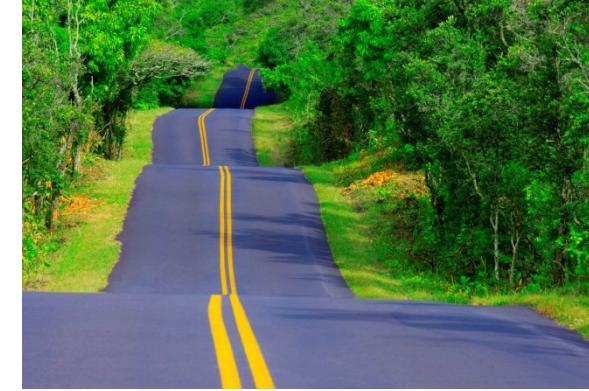
ISPs obtain IP addresses from a

- Local Internet Registry (LIR) or National Internet Registry (NIR),
- their appropriate Regional Internet Registry (RIR, 5 worldwide).



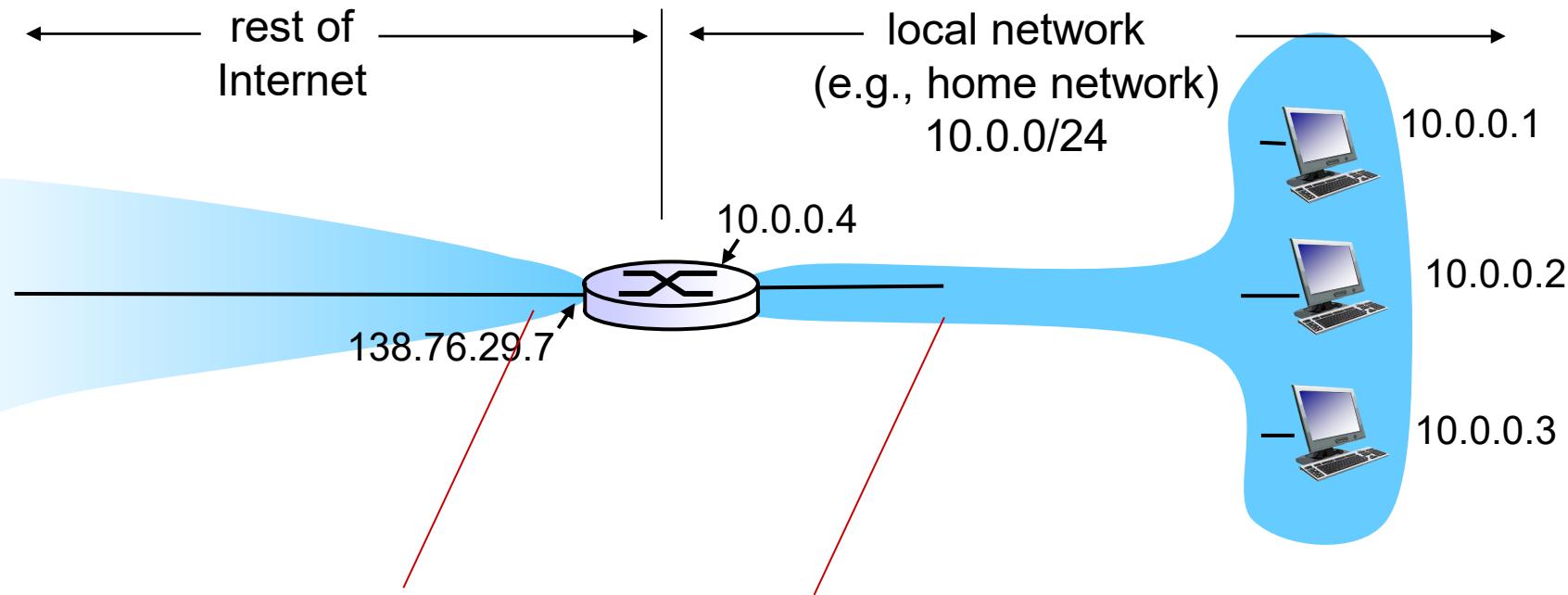
Roadmap Network Layer

- Forwarding versus routing
- Network layer service models
- Inside a router
- The Internet Network layer: IP, Addressing & related
 - Hierarchical addressing
 - Destination-based forwarding
 - How to get addresses
 - Well, it not really last word: ICANN allocated last chunk of IPv4 addresses to RRs in 2011!
 - NAT
 - IPv6
- (Next) Control, routing
 - path selection
 - instantiation, implementation in the Internet



NAT: network address translation

It is all about **extending the IP address space through MUX using the router's port#s** (it also “hides” addresses)



all datagrams *leaving* local network have *same* single source NAT IP address:
138.76.29.7, *different* source port numbers

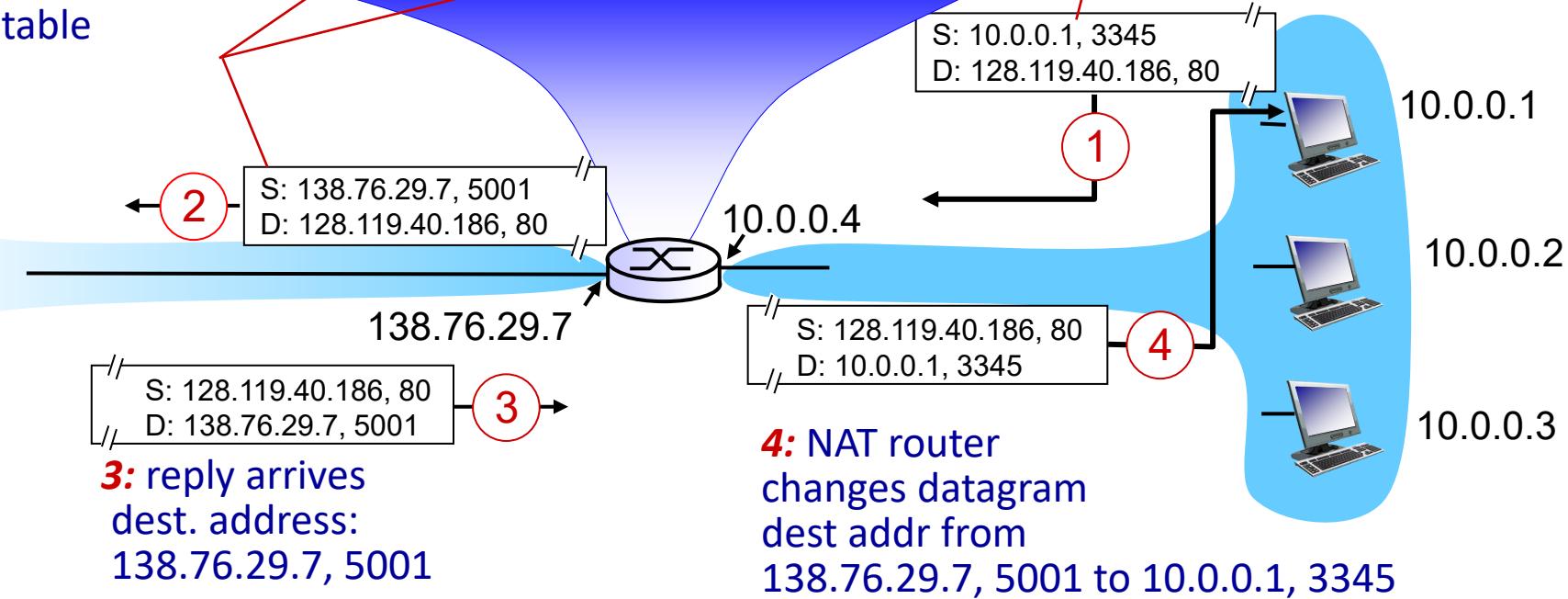
datagrams with source or destination in this network have 10.0.0/24 address for source, destination (as usual)

NAT: network address translation

2: NAT router changes datagram source addr from 10.0.0.1, 3345 to 138.76.29.7, 5001, updates table

NAT translation table	
WAN side addr	LAN side addr
138.76.29.7, 5001	10.0.0.1, 3345
.....

1: host 10.0.0.1 sends datagram to 128.119.40.186, 80



NAT: network address translation

- 16-bit port-number field:
 - 64k simultaneous connections within a single IP address!
- NAT is controversial:
 - routers should in principle process up to layer 3
 - violates end-to-end argument
 - NAT possibility must be taken into account by app designers, e.g., P2P applications
- but NAT is here to stay:
 - extensively used in home and institutional nets, 4G/5G cellular nets

Roadmap Network Layer

- Forwarding versus routing
- Network layer service models
- Inside a router
- The Internet Network layer: IP, Addressing related
 - Hierarchical addressing
 - Destination-based forwarding
 - How to get addresses
 - NAT
 - IPv6
- (Next) Control, routing
 - path selection
 - instantiation, implementation in the Internet



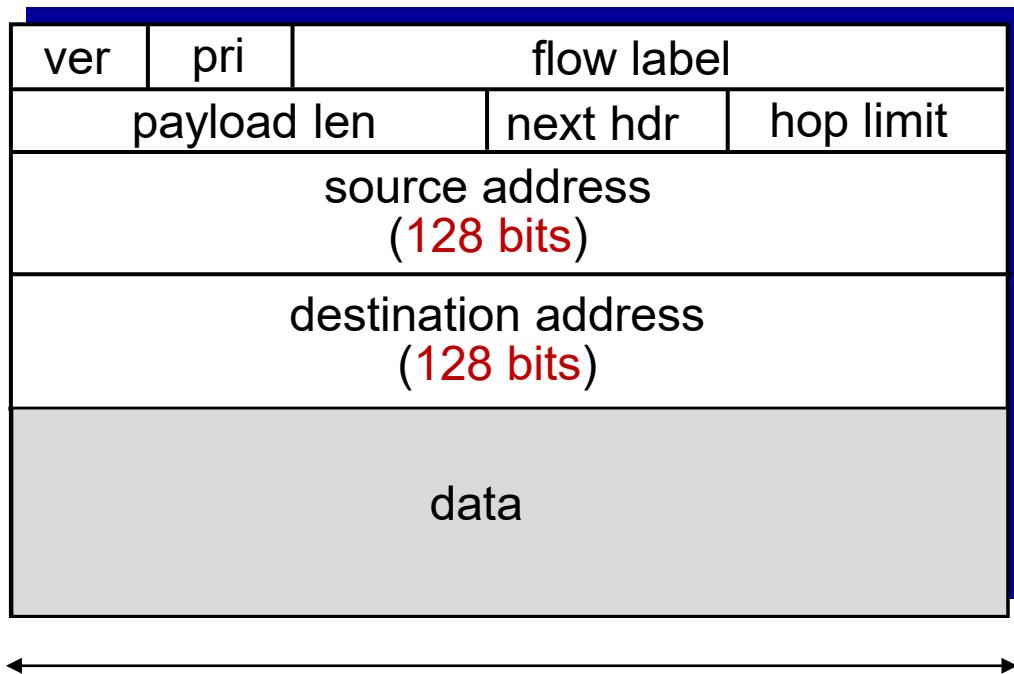
IPv6: motivation

- **initial motivation:** 32-bit address space almost completely allocated.
- **additional motivation:** header format must help to:
 - speed processing/forwarding
 - distinguishing types of traffic/flows (for potentially different services)

IPv6 datagram format

- fixed-length 40 byte header
- *128-bit addresses* ($2^{128} = 10^{38}$ hosts)
- standard subnet size: 2^{64} hosts
- no fragmentation allowed

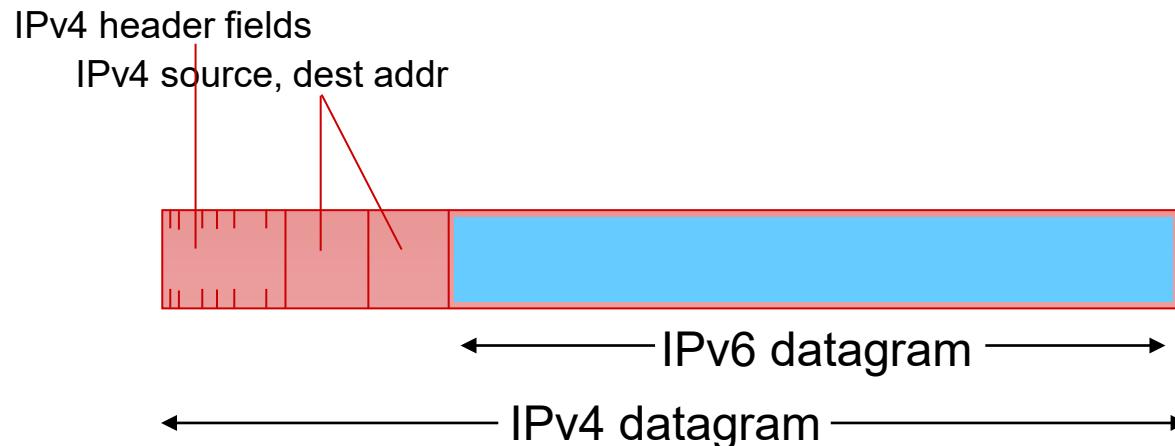
priority: identify priority among datagrams in flow
flow label: identify datagrams in same “flow.”
(concept of “flow” not well defined).



checksum: removed to reduce processing time
options: allowed, but outside of header, through “Next Header” field

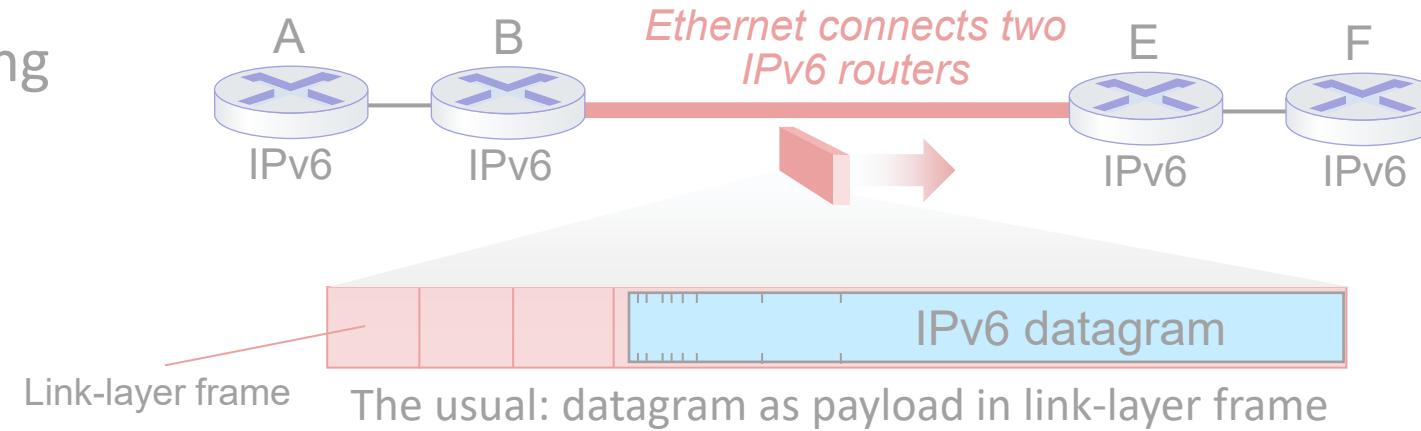
Transition from IPv4 to IPv6

- not all routers can be upgraded simultaneously
 - how can the network operate with mixed IPv4 and IPv6 routers?
- *tunneling*: IPv6 datagram carried as *payload* in IPv4 datagram among IPv4 routers
 - also used extensively in other contexts (4G/5G)

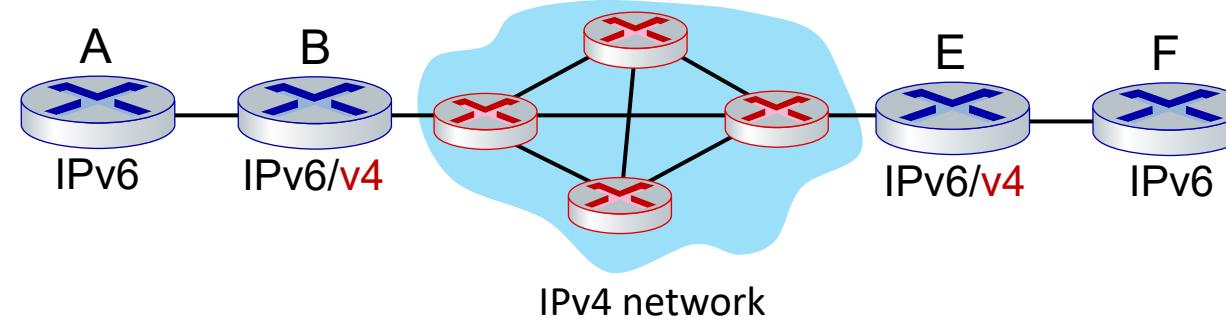


Tunneling and encapsulation

Ethernet connecting
two IPv6 routers:

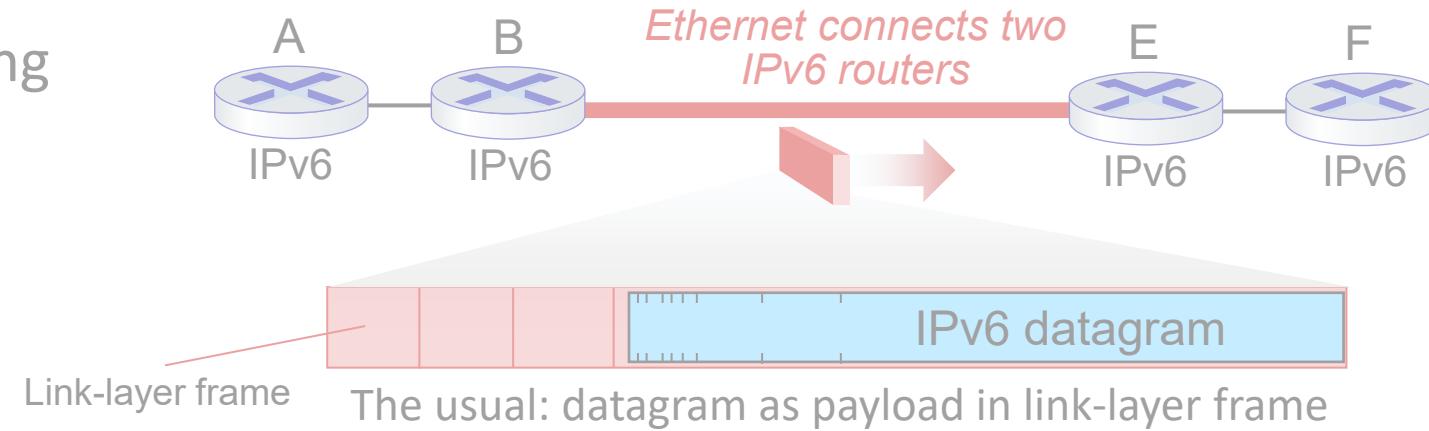


IPv4 network
connecting two
IPv6 routers

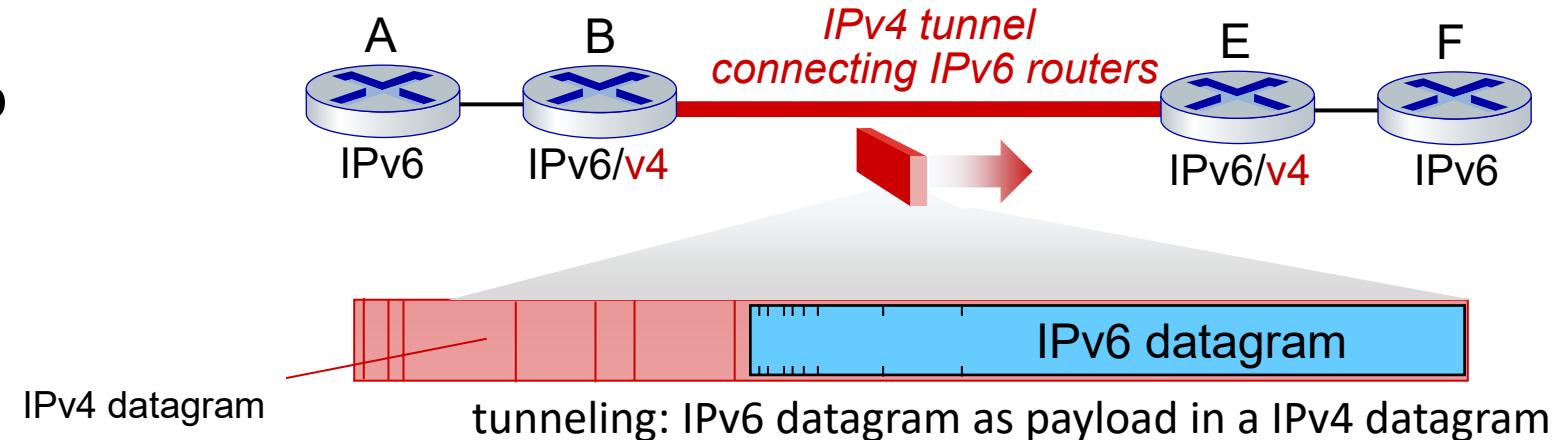


Tunneling and encapsulation

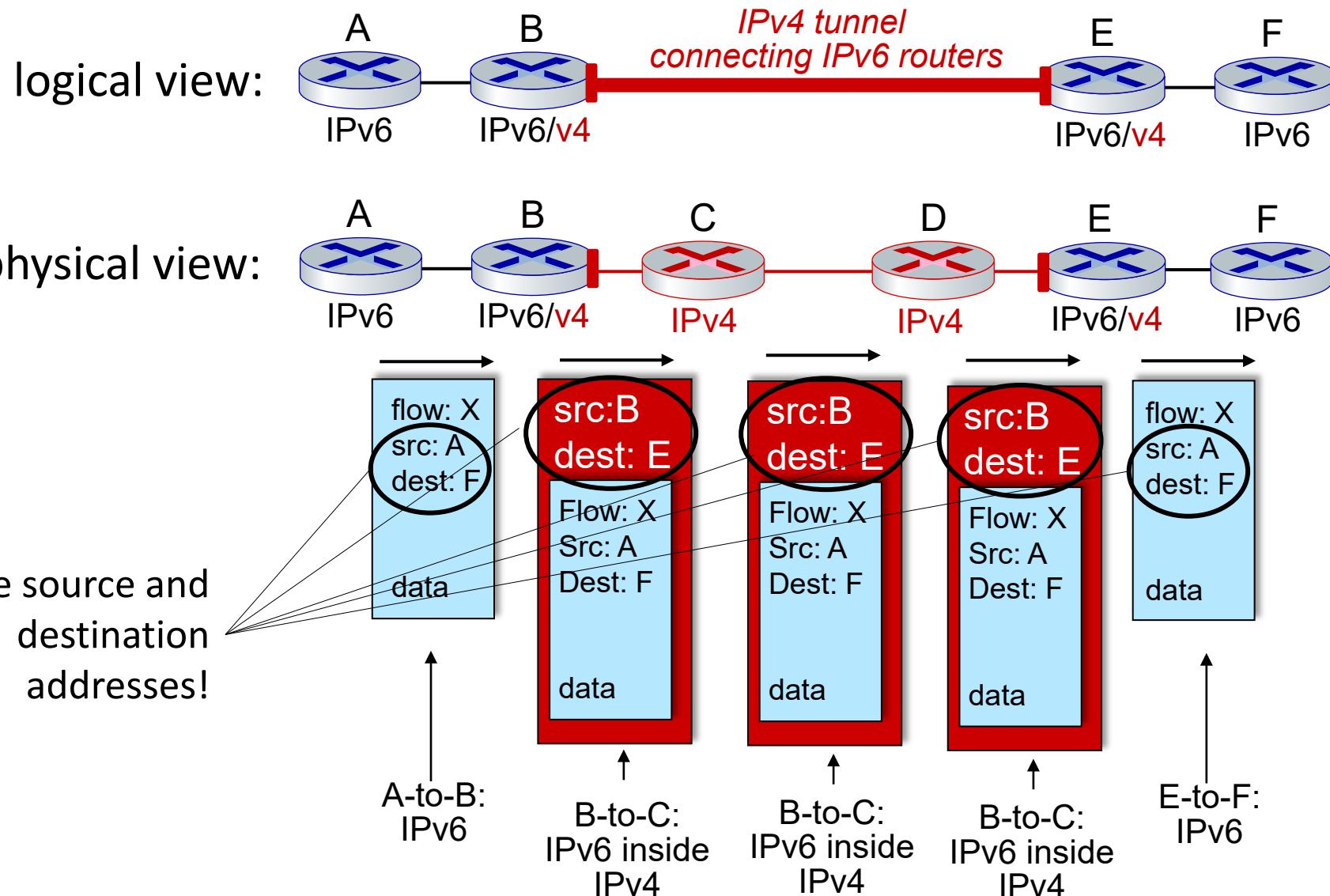
Ethernet connecting
two IPv6 routers:



IPv4 tunnel
connecting two
IPv6 routers



Tunneling



IPv6: adoption

- Google(*): 45% of clients access services via IPv6
- NIST: 1/3 of all US government domains are IPv6 capable
- *Long (long!) time for deployment, use*
25 years and counting!
 - think of application-level changes in last 25 years: www, streaming media, social media, gaming, teleconf, ...
 - Why?*

(*)<https://www.google.com/intl/en/ipv6/statistics.html>

Roadmap Network Layer

- Forwarding versus routing
- Network layer service models
 - Network layer architecture (shift): Software-Defined Networks
- How a router works
- The Internet Network layer: IP, Addressing & related
 - Hierarchical addressing
 - How to get addresses
 - Destination-based forwarding
 - NAT
 - IPv6
- **(Next) Control, routing**
 - path selection
 - instantiation, implementation in the Internet



- **KuroseRoss book**

Careful	Quick
4.1-4.3, 5.2-5.4, 5.6	
<i>SDN, data and control plane 4.4, 5.5: in subsequent lectures, connecting to multimedia/streaming</i>	5.7

- network layer **service models**
 - Contrast virtual circuit and datagram routing (simplicity, cost, purposes, what service types they may enable)
- **forwarding** versus **routing**
 - Explain the interplay between routing and forwarding
- how a **router works**
 - What is inside a router? How/where do queueing delays happen inside a router? Where/why can packets be dropped at a router?
- What is subnet? What is subnet masking?
 - Practice masking calculations, dividing address spaces
- Explain how to get an IP packet from source to destination
- Explain how NAT works.

Extra notes – Reflections, examples

Reflections on best-effort service:

- simplicity of mechanism has allowed Internet to be widely deployed adopted
- sufficient provisioning of bandwidth allows performance of real-time applications (e.g., interactive voice, video) to be “good enough” for “most of the time”
- replicated, application-layer distributed services (datacenters, content distribution networks) connecting close to clients’ networks, allow services to be provided from multiple locations
- congestion control of “elastic” services helps a lot!

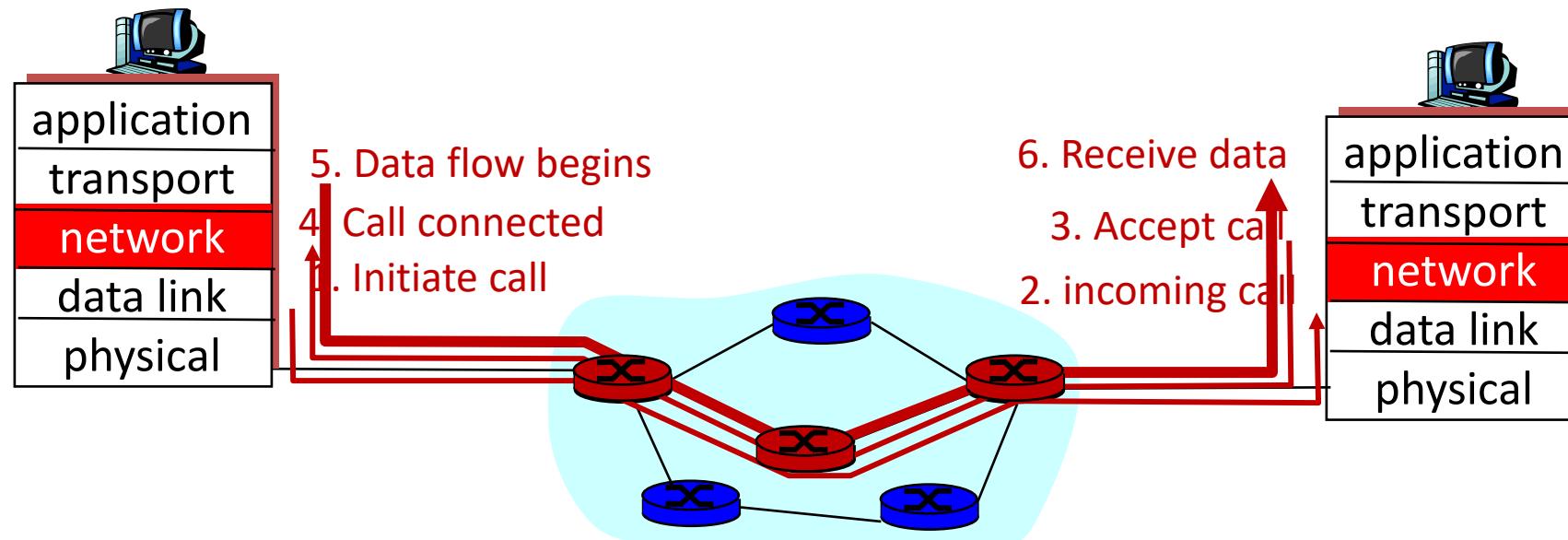
Hard to argue with success of best-effort service model ☺

Virtual circuits:

“source-to-dest path behaves *almost* like analog telephone circuit”

- call setup, teardown for each call *before* data can flow
 - signaling protocols to setup, maintain, teardown VC (ATM^(*), frame-relay, X.25; not in IP)
- each packet carries VC identifier (not destination host)
- resources (bandwidth, buffers) may be *allocated* to VC (dedicated resources = predictable service)

(*)Asynchronous Transfer Mode



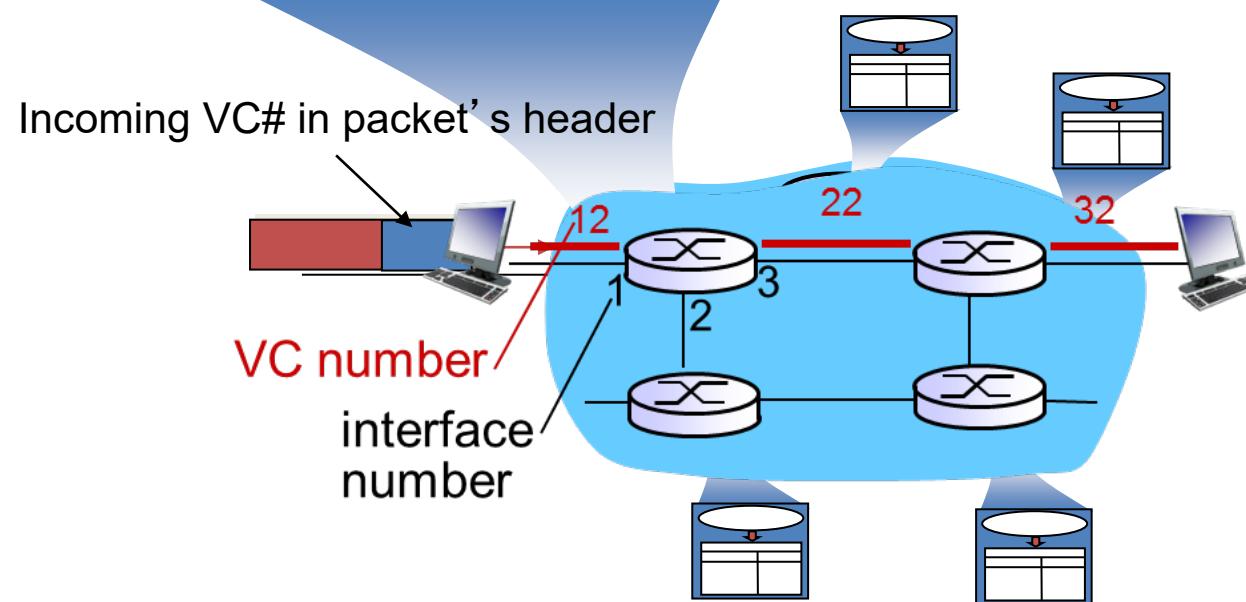
Virtual Circuits forwarding table

routing algorithm

local forwarding table

Incoming interface	Incoming VC #	Outgoing interface	Outgoing VC #
1	12	3	22
2	63	1	18
3	7	2	17
1	97	3	87
...

VC routers must maintain connection **state** information!



Network layer service models:

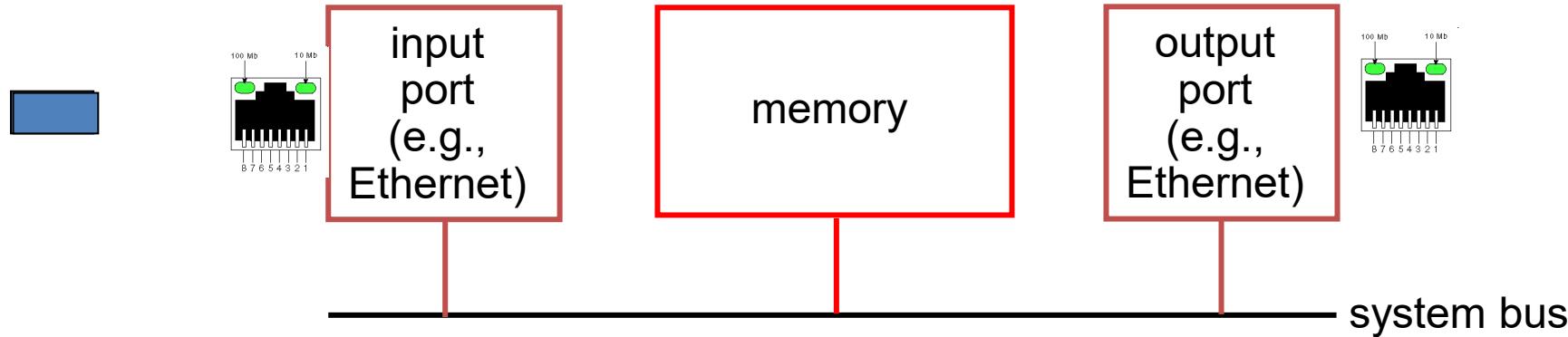
Network Architecture	Service Model	Guarantees ?				Congestion feedback
		Bandwidth	Loss	Order	Timing	
Internet	best effort	none	no	no	no	no (inferred via loss)
ATM	CBR	constant rate	yes	yes	yes	no congestion
ATM	VBR	guaranteed rate	yes	yes	yes	no congestion
ATM	ABR	guaranteed minimum	no	yes	no	yes
ATM	UBR	none	no	yes	no	no

- Internet models being extented
- (will study these later on)

Switching via memory

first generation routers:

- traditional computers with switching under direct control of CPU
- packet copied to system's memory
- speed limited by memory bandwidth (2 bus crossings per datagram)

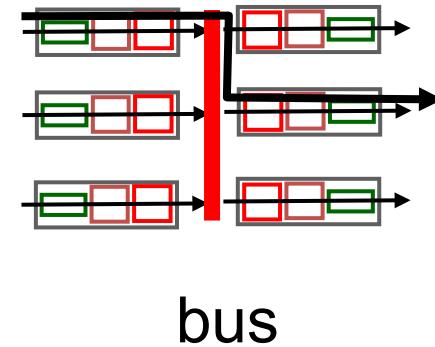


Switching via a bus

datagram from input port memory

to output port memory via a shared bus

- ***bus contention***: switching speed limited by bus bandwidth
- 32 Gbps bus, Cisco 5600: sufficient speed for access and enterprise routers



Some examples routers

Cisco Catalyst 3750E

Stackable (can combine units)
64 Gbps bandwidth
13 Mpps (packets per second)
12,000 address entries

Price: from 100 kSEK



HP ProCurve 6600-24G-4XG Switch

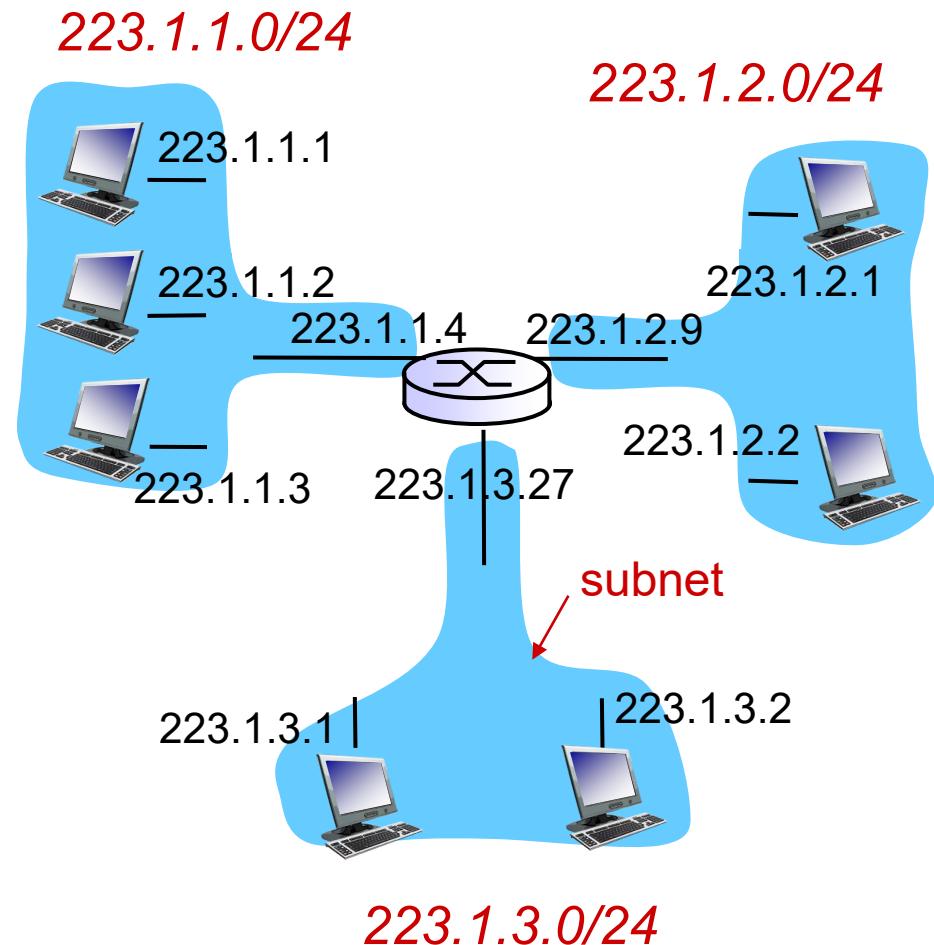
10 Gbps
Up to 75 Mpps (64-byte packets)
Latency: < 2.4 µs (FIFO 64-byte packets)
10,000 address entries

Price approx. 50 kSEK

Subnets

recipe

- to determine the subnets: detach each interface from its host or router, i.e. create islands of isolated networks
- each isolated network is a *subnet*



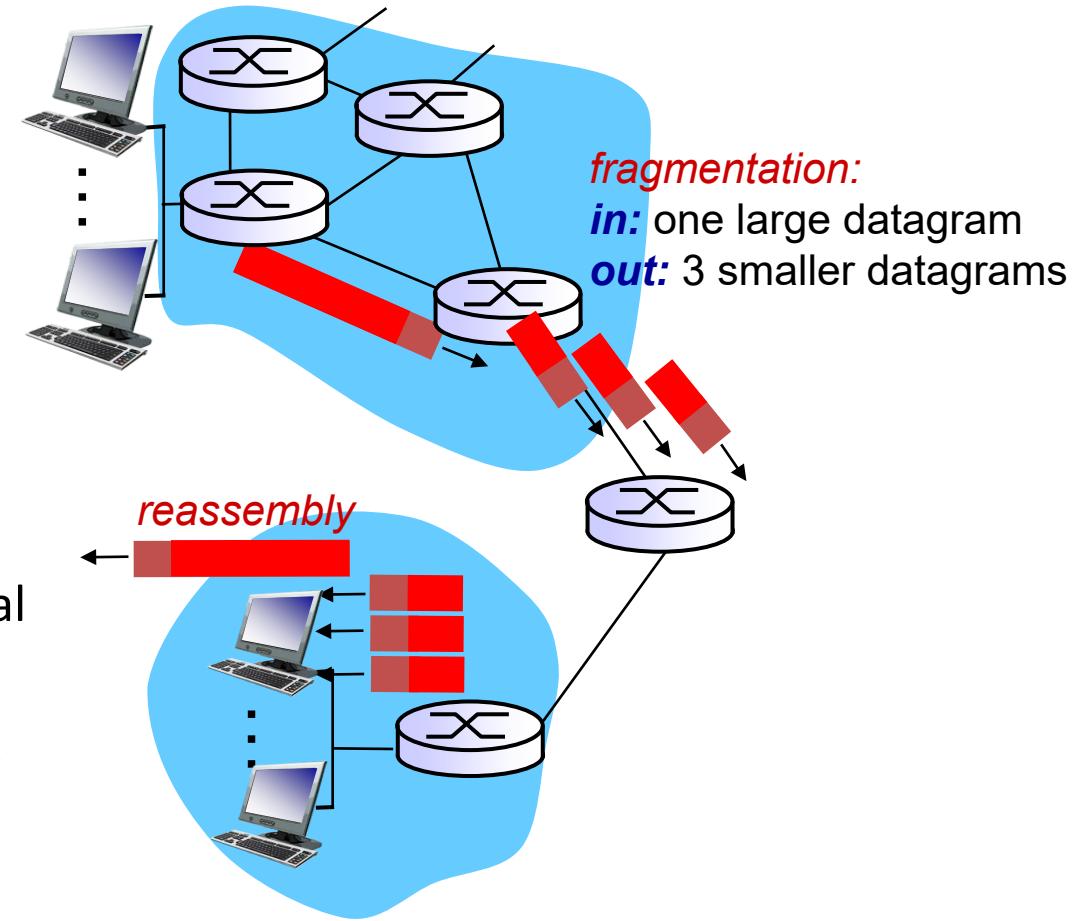
subnet mask: eg /24
defines how to find the subnet part of the address ...

CIDR Address Masks

<u>CIDR Notation</u>	<u>Dotted Decimal</u>	<u>CIDR Notation</u>	<u>Dotted Decimal</u>
/1	128.0.0.0	/17	255.255.128.0
/2	192.0.0.0	/18	255.255.192.0
/3	224.0.0.0	/19	255.255.224.0
/4	240.0.0.0	/20	255.255.240.0
/5	248.0.0.0	/21	255.255.248.0
/6	252.0.0.0	/22	255.255.252.0
/7	254.0.0.0	/23	255.255.254.0
/8	255.0.0.0	/24	255.255.255.0
/9	255.128.0.0	/25	255.255.255.128
/10	255.192.0.0	/26	255.255.255.192
/11	255.224.0.0	/27	255.255.255.224
/12	255.240.0.0	/28	255.255.255.240
/13	255.248.0.0	/29	255.255.255.248
/14	255.252.0.0	/30	255.255.255.252
/15	255.254.0.0	/31	255.255.255.254
/16	255.255.0.0	/32	255.255.255.255

IP fragmentation, reassembly

- network links have **MTU** (max.transfer size) - largest possible link-level frame
 - different link types, different MTUs
- large IP datagram divided (“fragmented”) within net
 - one datagram becomes several datagrams
 - “reassembled” only at final destination
 - IP header bits to identify + order related fragments



IP fragmentation, reassembly

example:

- ❖ 4000 byte datagram
- ❖ MTU = 1500 bytes

1480 bytes in
data field

offset =
 $1480/8$

	length =4000	ID =x	fragflag =0	offset =0	
--	-----------------	----------	----------------	--------------	--

*one large datagram becomes
several smaller datagrams*

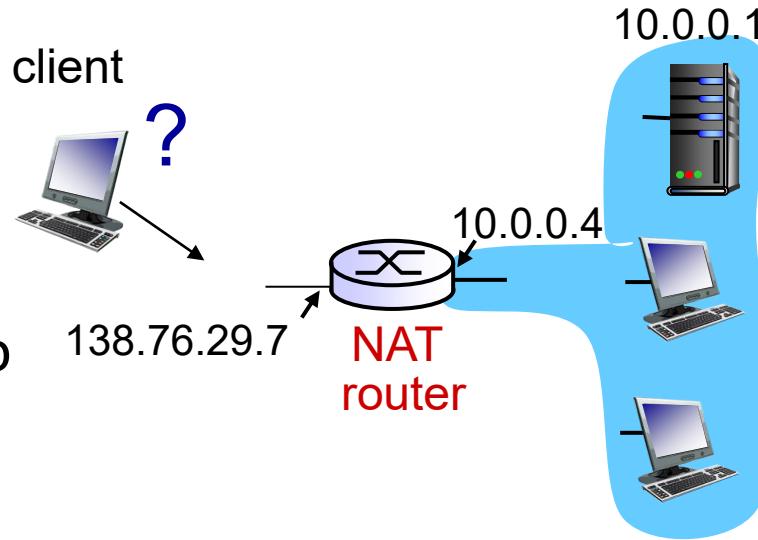
	length =1500	ID =x	fragflag =1	offset =0	
--	-----------------	----------	----------------	--------------	--

	length =1500	ID =x	fragflag =1	offset =185	
--	-----------------	----------	----------------	----------------	--

	length =1040	ID =x	fragflag =0	offset =370	
--	-----------------	----------	----------------	----------------	--

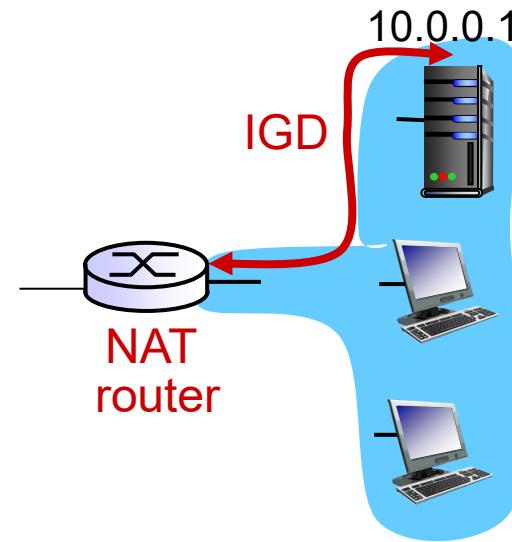
NAT traversal problem

- client wants to connect to server with address 10.0.0.1
 - server address 10.0.0.1 local to LAN (client can't use it as destination addr)
 - only one externally visible address: 138.76.29.7
- *solution 1*: statically configure NAT to forward incoming connection requests at given port to server
 - e.g., (123.76.29.7, port 2500) always forwarded to 10.0.0.1 port 25000
- *Solution 2*: automate the above through a protocol (universal plug-and-play)
- *Solution 3*: through a proxy/relay



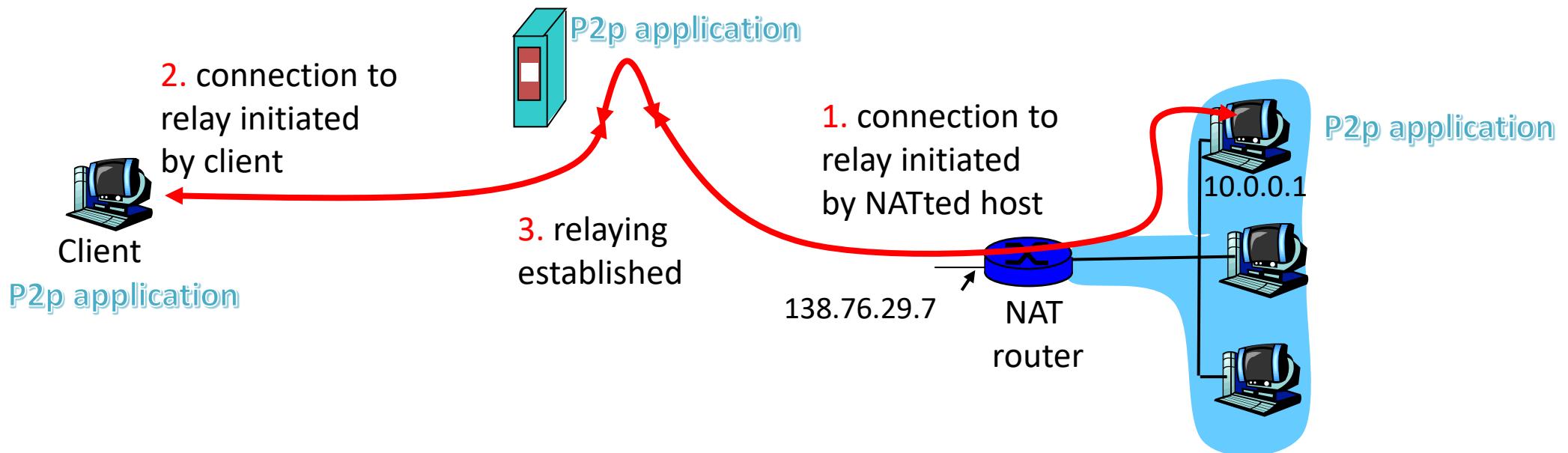
NAT traversal problem

- *solution 2:* Universal Plug and Play (UPnP) Internet Gateway Device (IGD) Protocol. Allows NATed host to:
 - ❖ learn public IP address (138.76.29.7)
 - ❖ add/remove port mappings (with lease times)i.e., automate static NAT port map configuration



NAT traversal problem

- solution 3 (application): relaying
 - NATed server establishes connection to relay
 - External client connects to relay
 - relay bridges packets between two connections



Getting a datagram from source to dest.

Getting a datagram from source to dest.

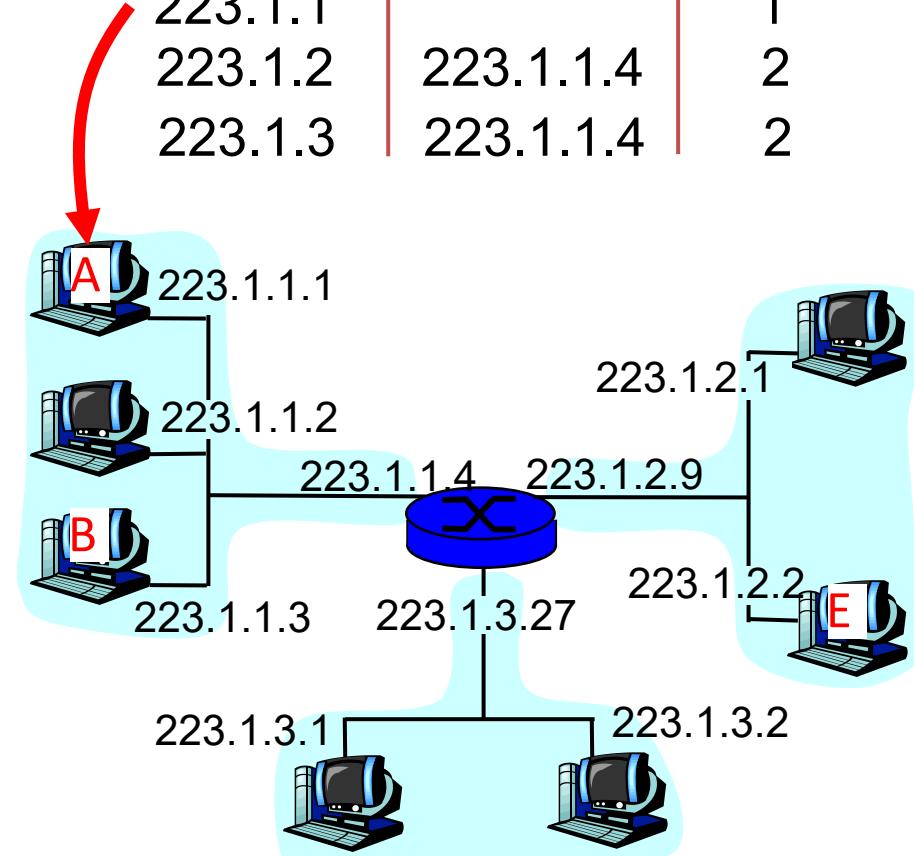
IP datagram: from A to B

misc fields	source IP addr	dest IP addr	data
-------------	----------------	--------------	------

- datagram remains unchanged, as it travels source to destination
- addr fields of interest here

forwarding table in A

Dest. Net.	next router	Nhops
223.1.1		1
223.1.2	223.1.1.4	2
223.1.3	223.1.1.4	2



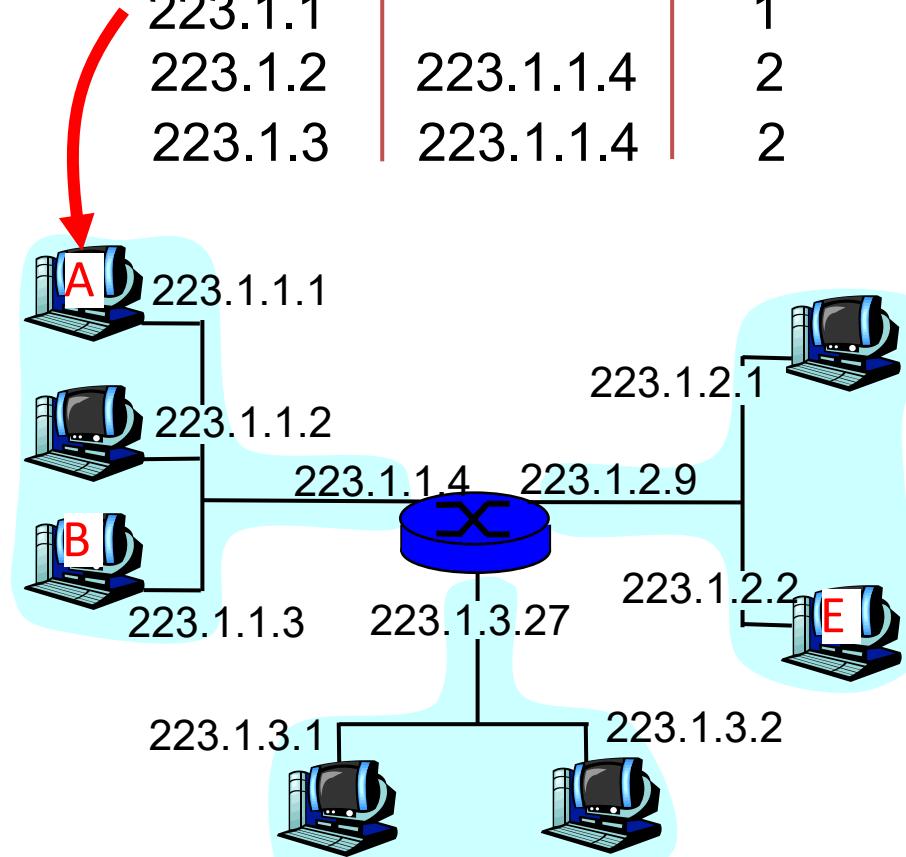
Getting a datagram from source to dest.

misc fields	223.1.1.1	223.1.1.3	data
-------------	-----------	-----------	------

Starting at A, given IP datagram addressed to B:

- look up net. address of B
- find B is on **same net.** as A (B and A are directly connected)
- link layer** will send datagram directly to B (inside link-layer frame)

Dest. Net.	next router	Nhops
223.1.1		1
223.1.2	223.1.1.4	2
223.1.3	223.1.1.4	2



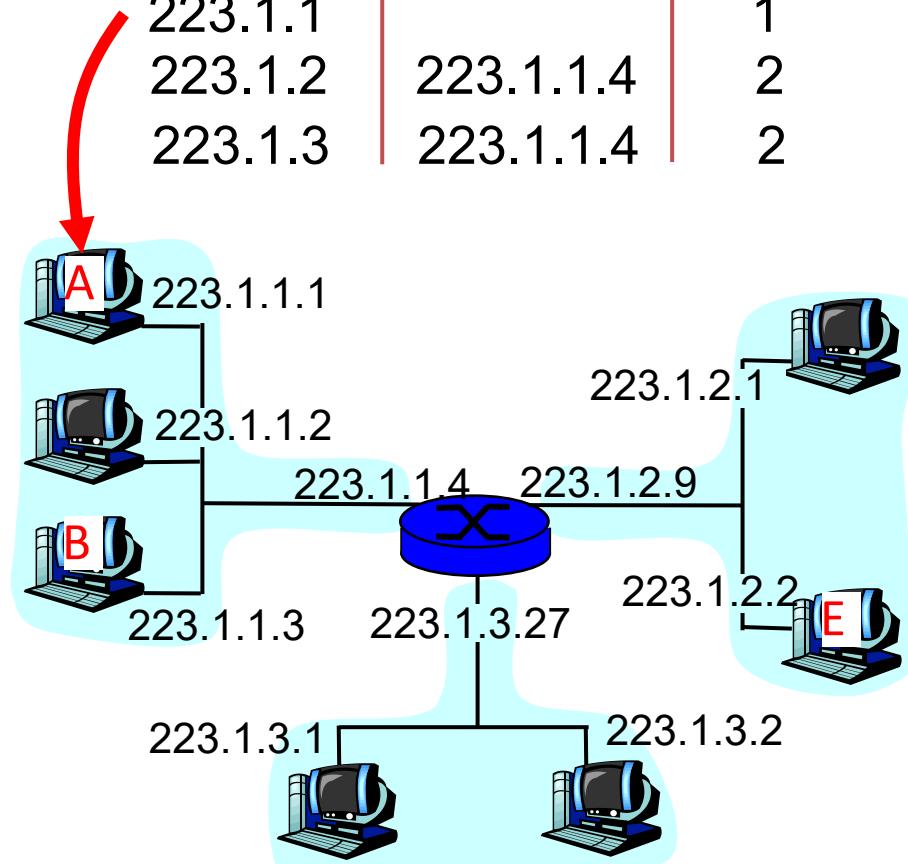
Getting a datagram from source to dest.

misc fields	223.1.1.1	223.1.2.3	data
-------------	-----------	-----------	------

Now: starting at A, dest. E:

- look up network address of E
- E on *different network*
- routing table: next hop router to E is 223.1.1.4
- link layer** is asked to send datagram to router 223.1.1.4 (inside link-layer frame)
- datagram arrives at 223.1.1.4
- continued....

Dest. Net.	next router	Nhops
223.1.1		1
223.1.2	223.1.1.4	2
223.1.3	223.1.1.4	2



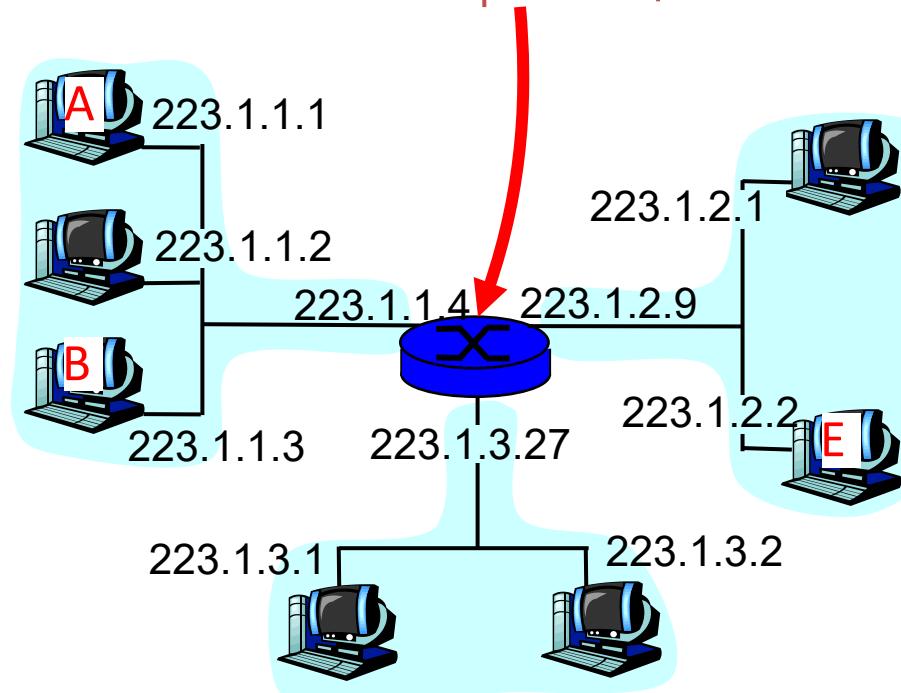
Getting a datagram from source to dest.

misc fields	223.1.1.1	223.1.2.3	data
-------------	-----------	-----------	------

Arriving at 223.1.4, destined for
223.1.2.2

- look up network address of E
- E on *same* network as router's interface 223.1.2.9
 - router, E directly attached
- link layer sends datagram to 223.1.2.2 (inside link-layer frame) via interface 223.1.2.9
- datagram arrives at 223.1.2.2!!!
(hooray!)

Dest. network	next router	Nhops	interface
223.1.1	-	1	223.1.1.4
223.1.2	-	1	223.1.2.9
223.1.3	-	1	223.1.3.27





**Become a student
representative
TODAY!**

Join after the lecture for the first feedback meeting.

Guest Lecture on Mon. 4th March at 10:00 **VOLVO**

- **Mahboobeh Daftari** is a Security Architect (PKI Lead Architect) at Volvo Trucks, Göteborg.
 - She has worked as Security Expert in the automotive industry for more than 6 years.
- She graduated from Chalmers in 2016 with a M.Sc. in Computer Systems and Networks



“Communication between vehicles and OEM backend systems is required for many purposes including incident analysis and vehicle software update. A common way to secure end-to-end communication between Electronic Control Units (ECUs) and OEM backend systems is to secure diagnostic service requests and responses between diagnostic tools and the ECUs.

Unified Diagnostic Services (UDS) protocol is a standard protocol defining two services for enabling security of diagnostics on the application layer: Security Access (0x27) and Authentication Service (0x29). The overall security of the communication depends on the implementation of the services and the underlying key management which is OEM specific.”



Course on Computer Communication and Networks

Lecture 7

Chapter 5; Network Layer: Control Plane

EDA344/DIT423/LEU062

Lecturer: Hans-Martin Heyn

Based on the book Computer Networking: A Top Down Approach, Jim Kurose, Keith Ross, Addison-Wesley.

Layering: The OSI Reference Model

ISO (International Standards Organization) defined the OSI (Open Systems Interconnect) model to help vendors create interoperable network implementation

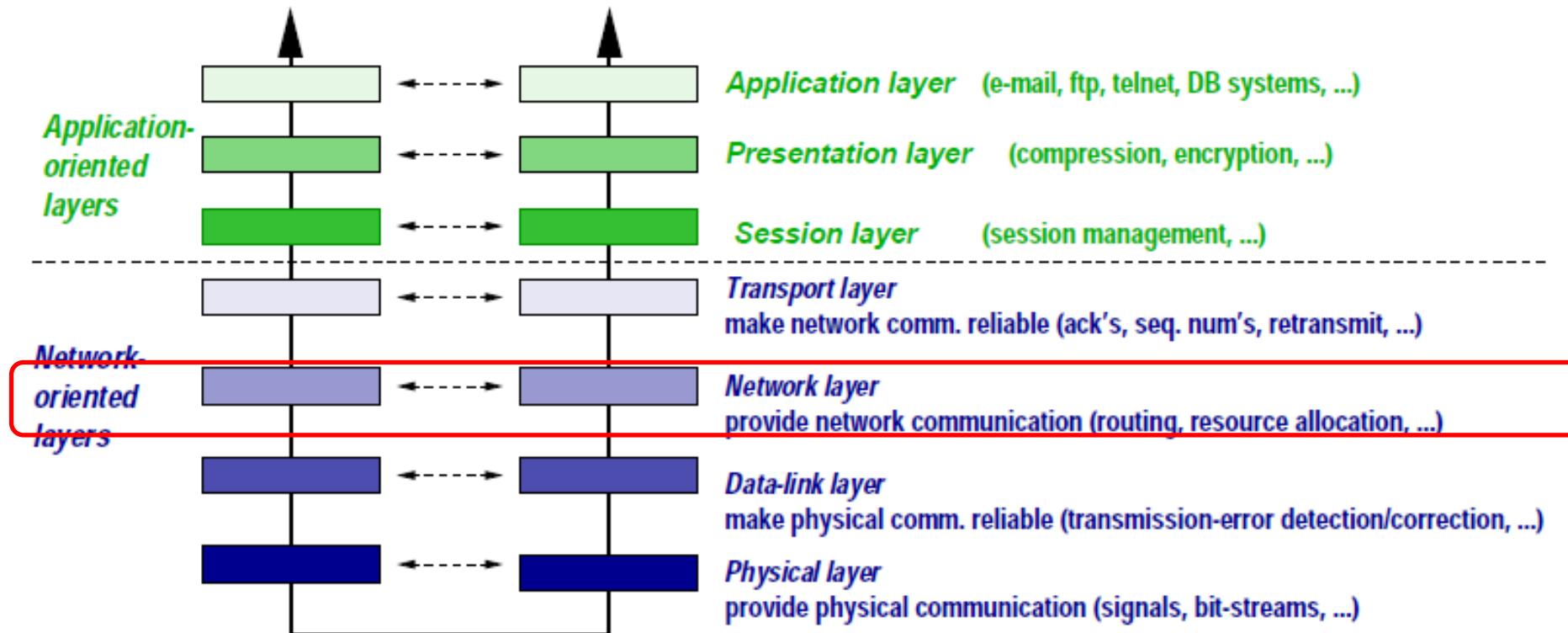


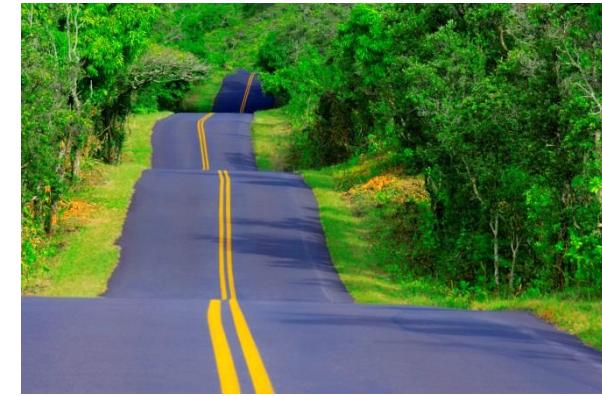
Fig. Steen, Sips : Computer and Network organization

``X dot'' series (X.25, X.400, X.500) OSI model implementation (protocol stack)

Roadmap Network Layer

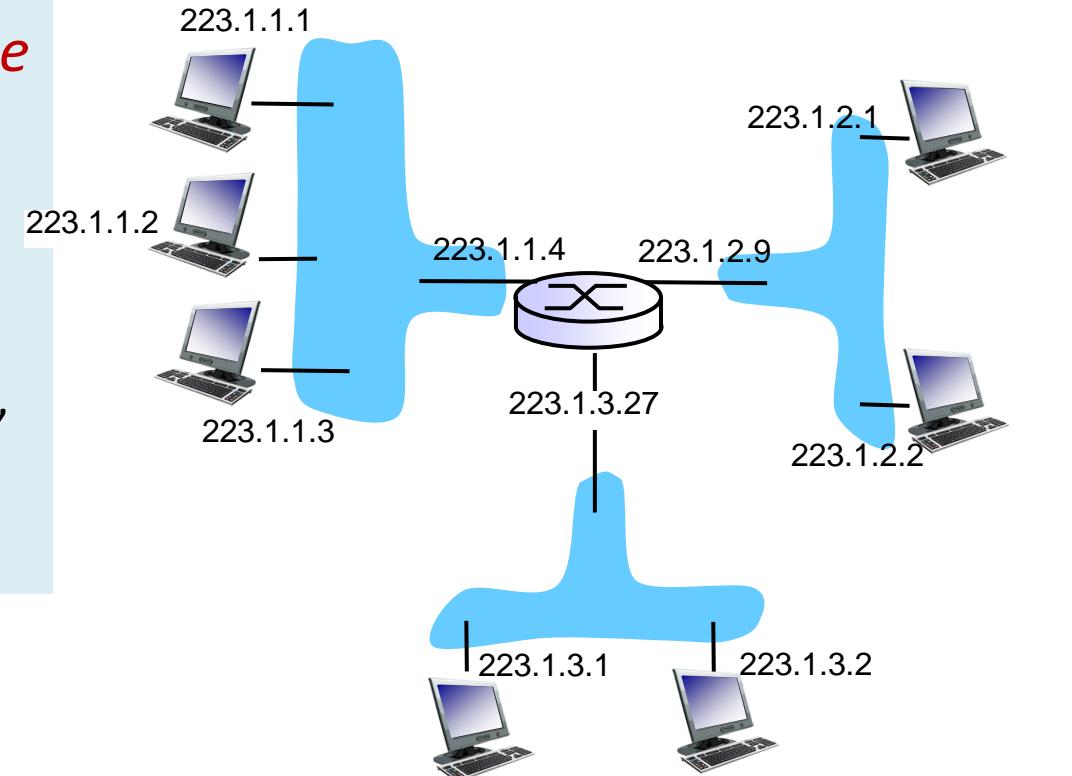


- Network layer: Data Plane
 - Forwarding vs routing
 - Network layer service models
 - Inside a router
 - The Internet Network layer: IP addressing beyond IPv4
- Network layer: Control Plane
 - Routing Protocols
 - Link State
 - Distance Vector
 - Intra-ISP routing: OSPF
 - Routing among ISPs: BGP
 - Internet Control Message Protocol



IP addressing: introduction

- **IP address:** 32-bit id for host/router *interface*
- **interface:** connection between host/router and physical link
 - routers have multiple interfaces
 - common end-host typically has 1-2 interfaces (e.g., wired Ethernet and wireless 802.11); servers can have more



$223.1.1.1 = \underline{11011111} \underline{00000001} \underline{00000001} \underline{00000001}$

223 1 1 1

IP addressing: CIDR

CIDR: Classless InterDomain Routing

- subnet portion of address of arbitrary length
- address format: **a.b.c.d/x**, where x is # bits in subnet portion of address



IP Addressing: address allocation

Q: How does an ISP get block of addresses?

A: ICANN: <http://www.icann.org/>

Internet Corporation for Assigned Names and Numbers



- allocates addresses
- manages DNS
- assigns domain names, resolves disputes

ISPs obtain IP addresses from a

- Local Internet Registry (LIR) or National Internet Registry (NIR),
- their appropriate Regional Internet Registry (RIR, 5 worldwide).



IP Addressing: are we running out of addresses?



$4.294.967.296 - 588.514.304$ (reserved)
= 3.706.452.992 addresses available

Private networks have designated address ranges:

RFC 1918 name	IP address range	Number of addresses	Largest CIDR block (subnet mask)	Host ID size	Mask bits	<i>Classful</i> description <small>[Note 1]</small>
24-bit block	10.0.0.0 – 10.255.255.255	16 777 216	10.0.0.0/8 (255.0.0.0)	24 bits	8 bits	single class A network
20-bit block	172.16.0.0 – 172.31.255.255	1 048 576	172.16.0.0/12 (255.240.0.0)	20 bits	12 bits	16 contiguous class B networks
16-bit block	192.168.0.0 – 192.168.255.255	65 536	192.168.0.0/16 (255.255.0.0)	16 bits	16 bits	256 contiguous class C networks

IP Addressing: IPv4 addresses as a scarce resource

Amazon Web Services (AWS) price table for IP4 addresses:

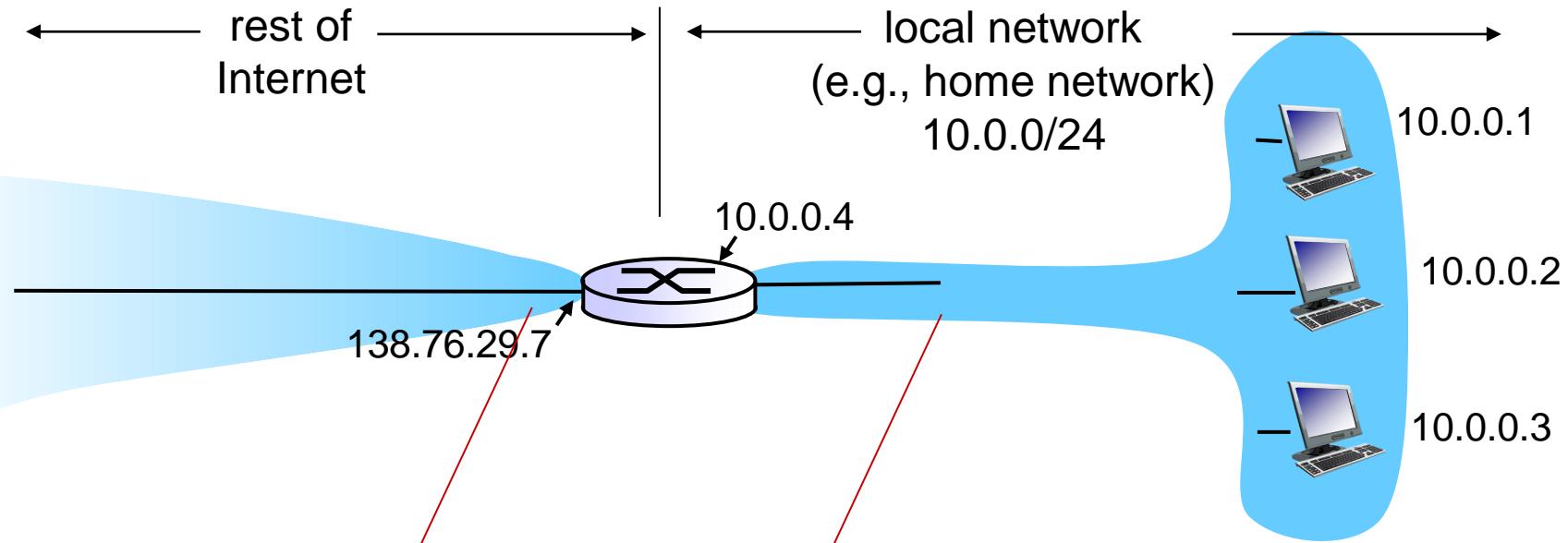
Public IPv4 Charge

As you may know, IPv4 addresses are an increasingly scarce resource and the cost to acquire a single public IPv4 address has risen more than 300% over the past 5 years. This change reflects our own costs and is also intended to encourage you to be a bit more frugal with your use of public IPv4 addresses and to think about accelerating your adoption of IPv6 as a modernization and conservation measure.

Public IP Address Type	Current Price/Hour (USD)	New Price/Hour (USD) (Effective February 1, 2024)
In-use Public IPv4 address (including Amazon provided public IPv4 and Elastic IP) assigned to resources in your VPC, Amazon Global Accelerator, and AWS Site-to-site VPN tunnel	No charge	\$0.005
Additional (secondary) Elastic IP Address on a running EC2 instance	\$0.005	\$0.005
Idle Elastic IP Address in account	\$0.005	\$0.005

NAT: network address translation

It is all about **extending the IP address space through MUX using the router's port#s** (it also “hides” addresses)



all datagrams *leaving* local network have *same* single source NAT IP address: 138.76.29.7, *different* source port numbers

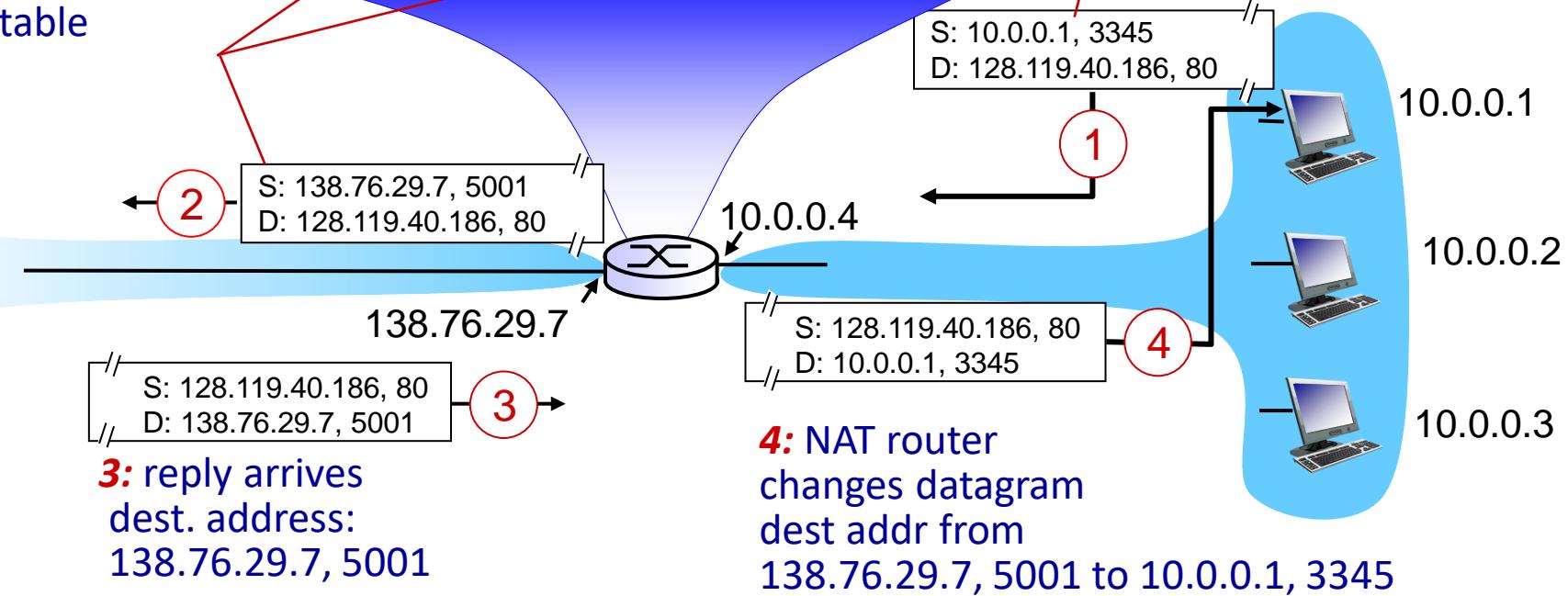
datagrams with source or destination in this network have 10.0.0/24 address for source, destination (as usual)

NAT: network address translation

2: NAT router changes datagram source addr from 10.0.0.1, 3345 to 138.76.29.7, 5001, updates table

NAT translation table	
WAN side addr	LAN side addr
138.76.29.7, 5001	10.0.0.1, 3345
.....

1: host 10.0.0.1 sends datagram to 128.119.40.186, 80



NAT: network address translation

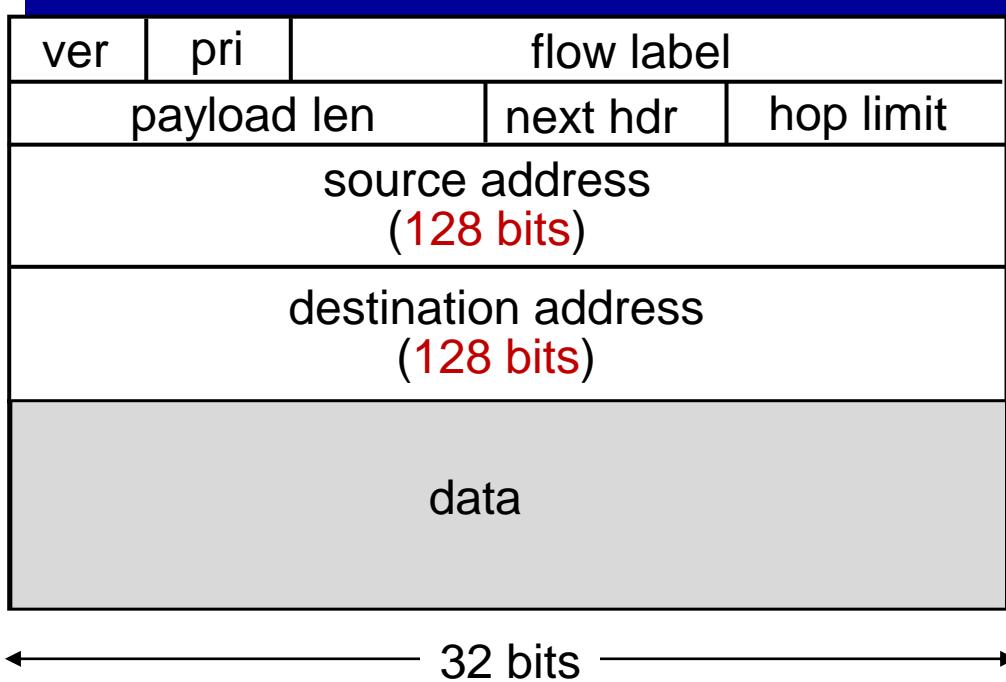
- 16-bit port-number field:
 - 64k simultaneous connections within a single IP address!
- NAT is controversial:
 - routers should in principle process up to layer 3
 - violates end-to-end argument
 - NAT possibility must be taken into account by app designers, e.g., P2P applications
- but NAT is here to stay:
 - extensively used in home and institutional nets, 4G/5G cellular nets

IPv6: motivation

- **initial motivation:** 32-bit address space almost completely allocated.
- **additional motivation:** header format must help to:
 - speed processing/forwarding
 - distinguishing types of traffic/flows (for potentially different services)

IPv6 datagram format

- fixed-length 40 byte header
- 128-bit addresses ($2^{128} = 10^{38}$ hosts)
 - IPv4: ($2^{32} = 10^9$ hosts)
 - ca. 10^{11} stars in Milky Way
 - ca. 10^{18} sand grains on Earth
- standard subnet size: $2^{64} = 10^{19}$ hosts



priority: identify priority among datagrams in flow

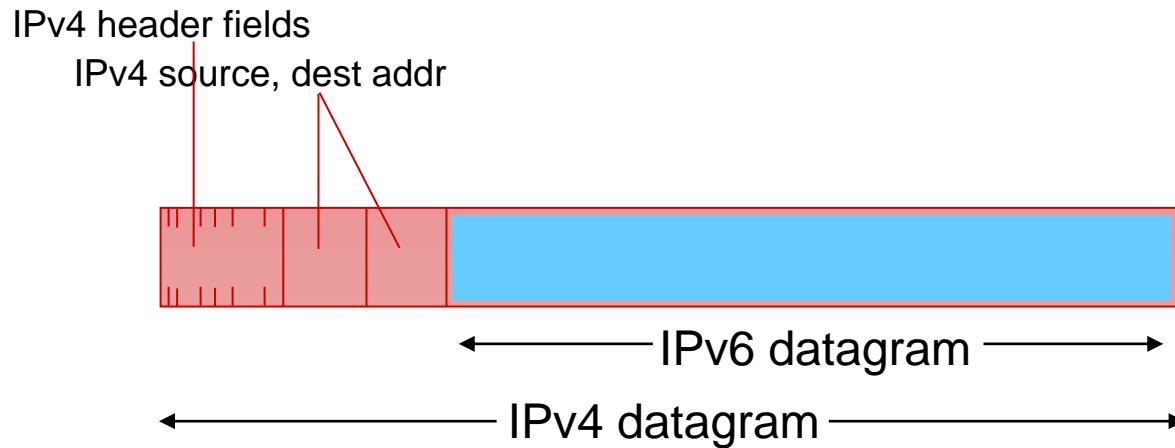
flow label: identify datagrams in same “flow.”
(concept of “flow” not well defined).

checksum: removed to reduce processing time

options: allowed, but outside of header, through
“Next Header” field

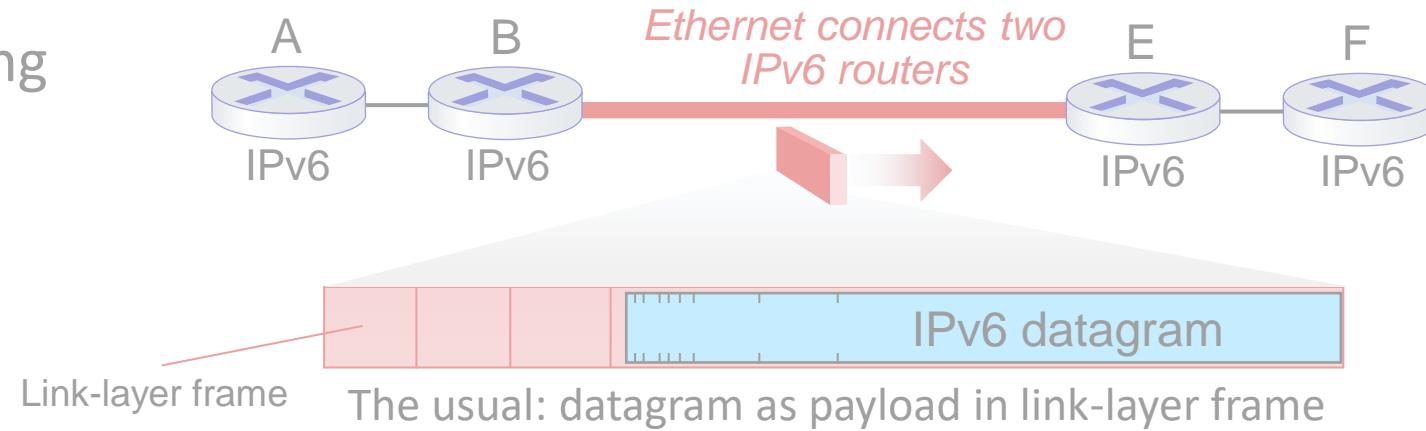
Transition from IPv4 to IPv6

- not all routers can be upgraded simultaneously
 - how can the network operate with mixed IPv4 and IPv6 routers?
- **tunneling:** IPv6 datagram carried as *payload* in IPv4 datagram among IPv4 routers
 - also used extensively in other contexts (4G/5G)

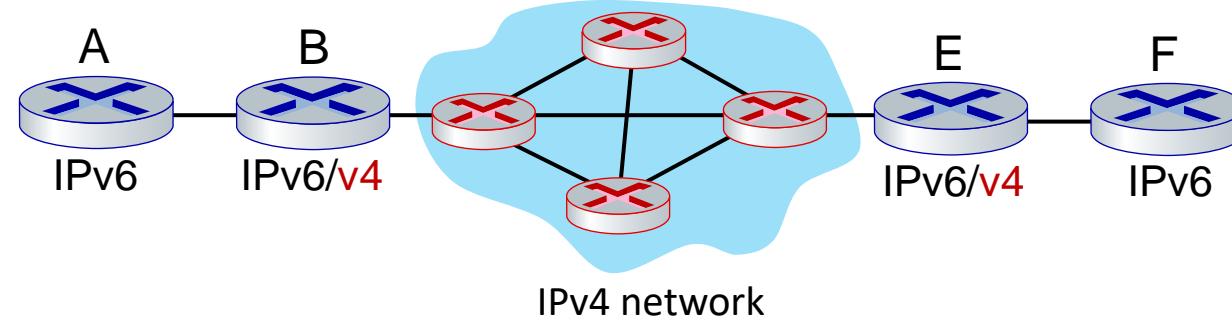


Tunneling and encapsulation

Ethernet connecting
two IPv6 routers:

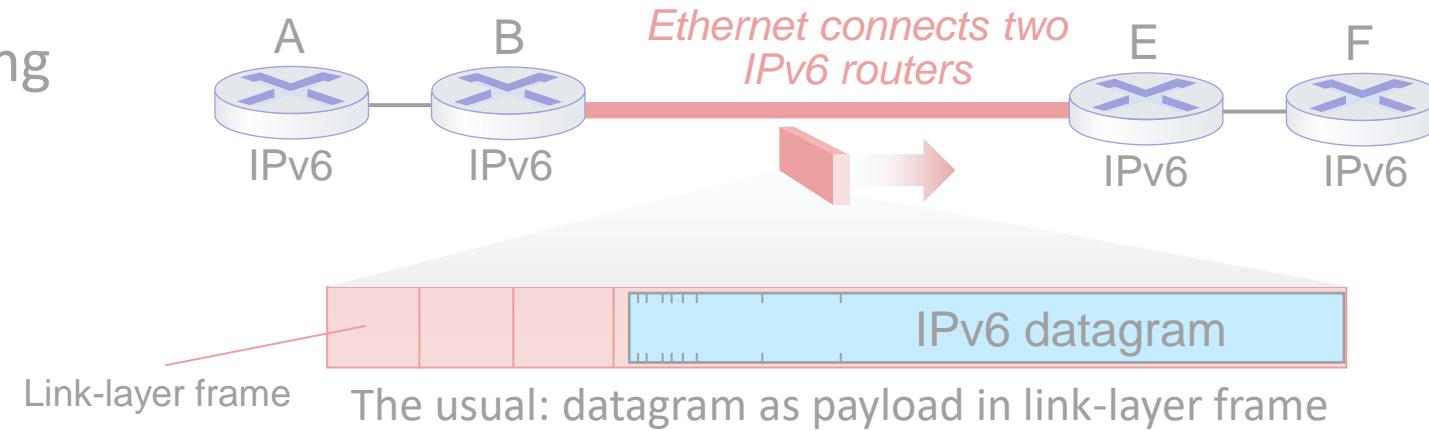


IPv4 network
connecting two
IPv6 routers

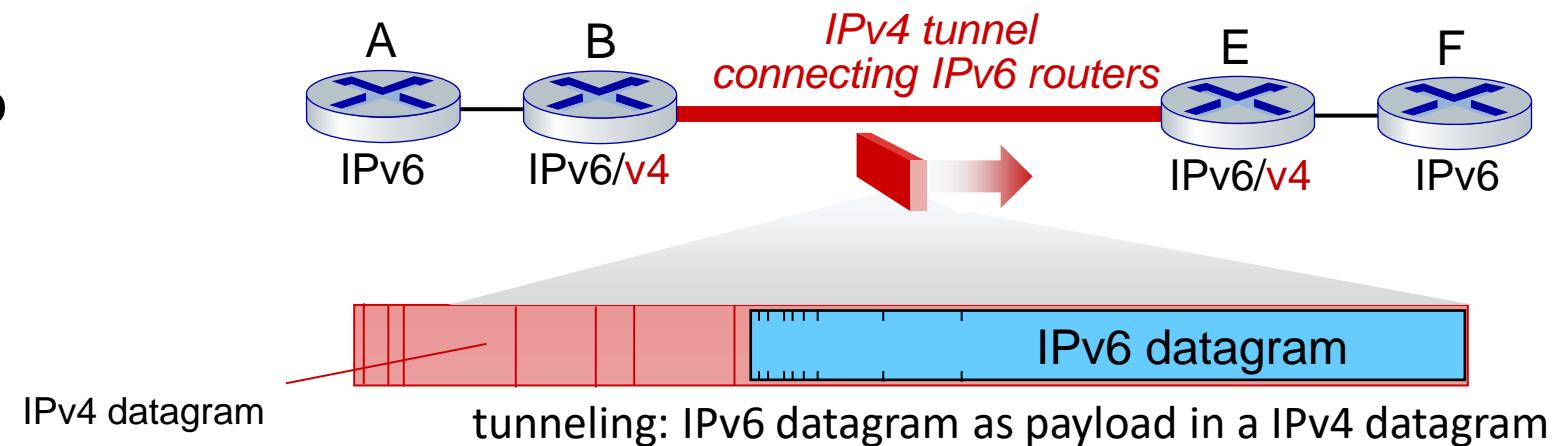


Tunneling and encapsulation

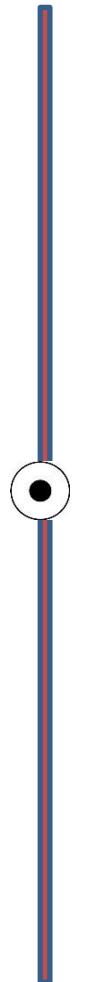
Ethernet connecting
two IPv6 routers:



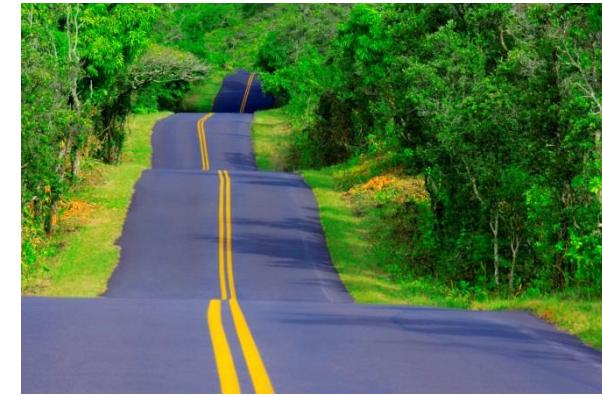
IPv4 tunnel
connecting two
IPv6 routers



Roadmap Network Layer

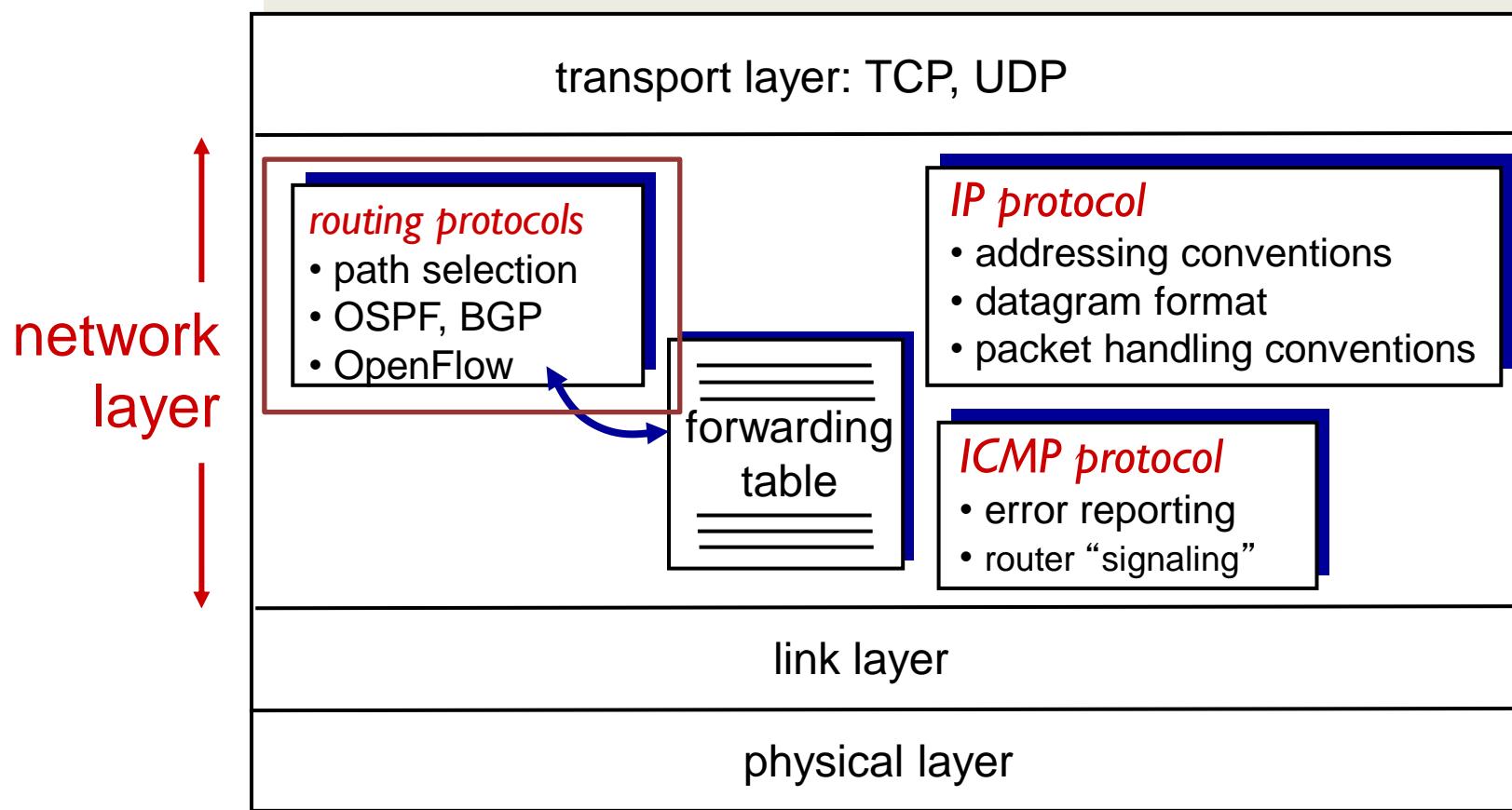


- Network layer: Data Plane
 - Forwarding vs routing
 - Network layer service models
 - Inside a router
 - The Internet Network layer: IP, Addressing & related
- Network layer: Control Plane
 - Routing Protocols
 - Link State
 - Distance Vector
 - Intra-ISP routing: OSPF
 - Routing among ISPs: BGP
 - Internet Control Message Protocol



The Internet network layer

host, router network layer functions:



Forwarding vs. Routing

- **forwarding:** move packets from router's input to appropriate router output
- **routing:** determine route taken by packets from source to destination

data plane

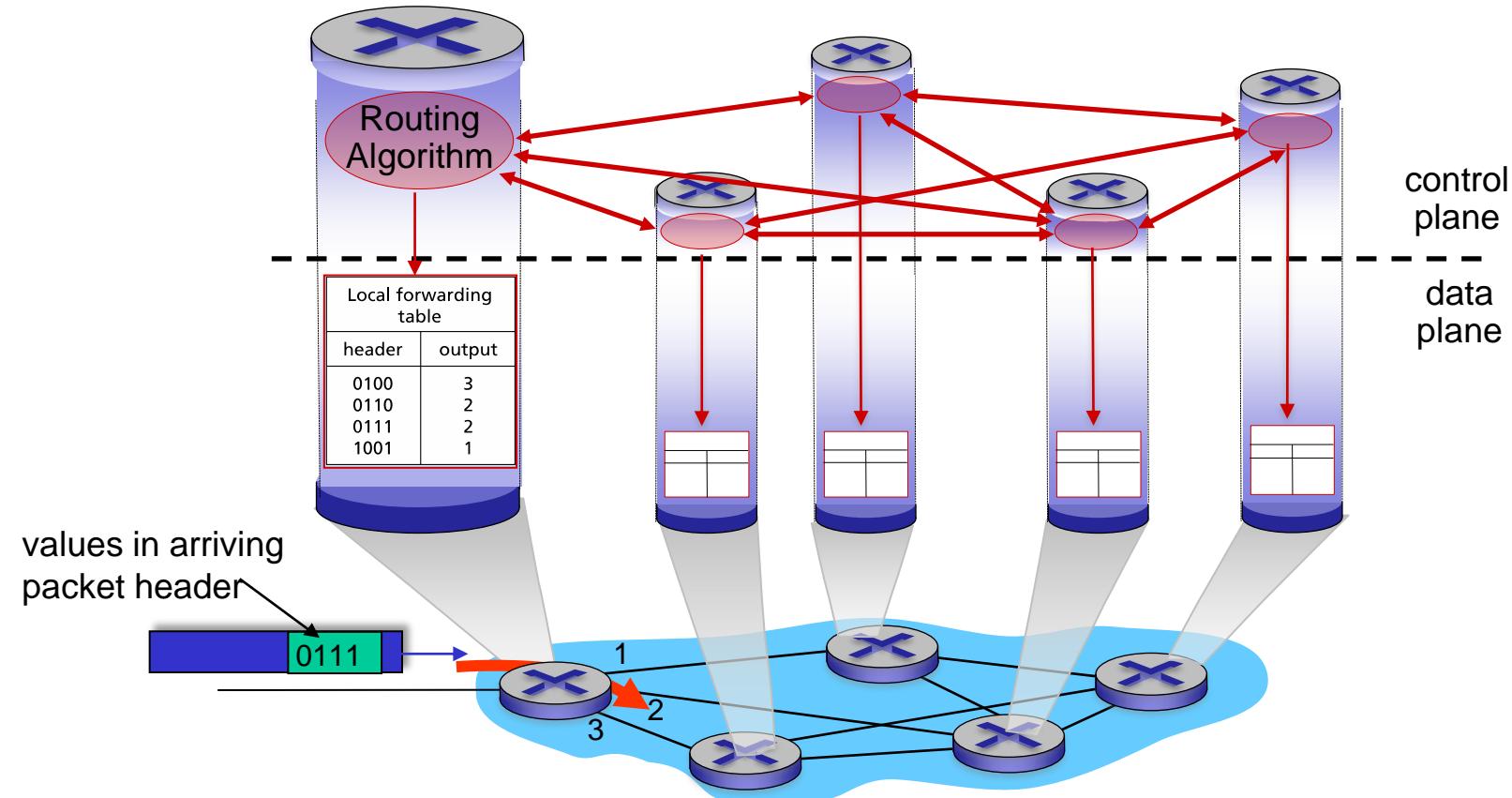
control plane

Two approaches to structuring network control plane:

- per-router control (traditional)
- logically centralized control (software defined networking)

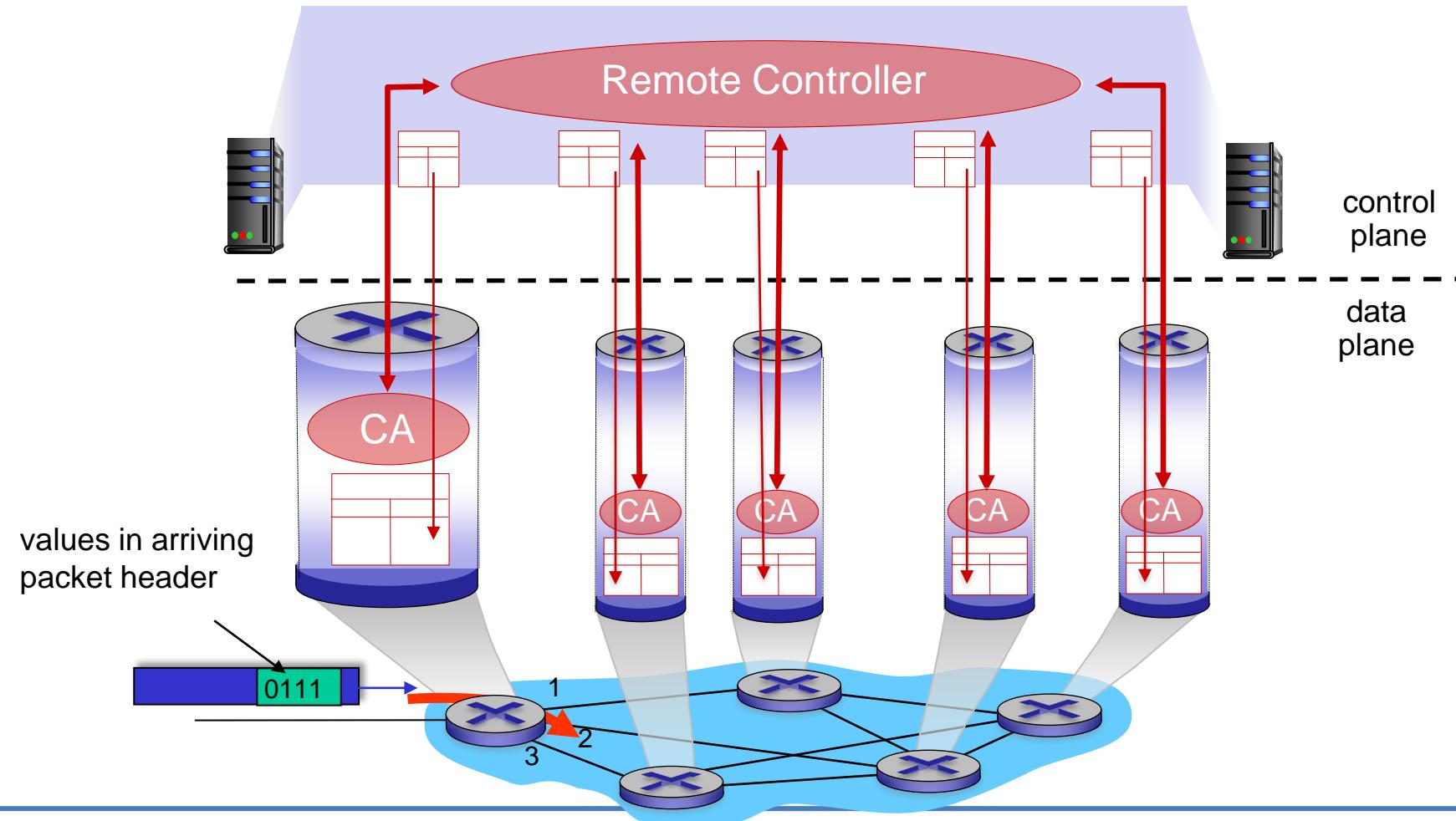
Per-router control plane

Individual routing algorithm components *in each and every router* interact in the control plane



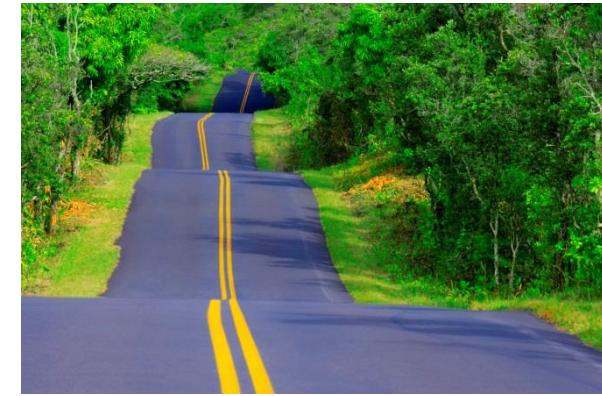
Software-Defined Networking (SDN) control plane

Remote controller computes, installs forwarding tables in routers



Roadmap Network Layer

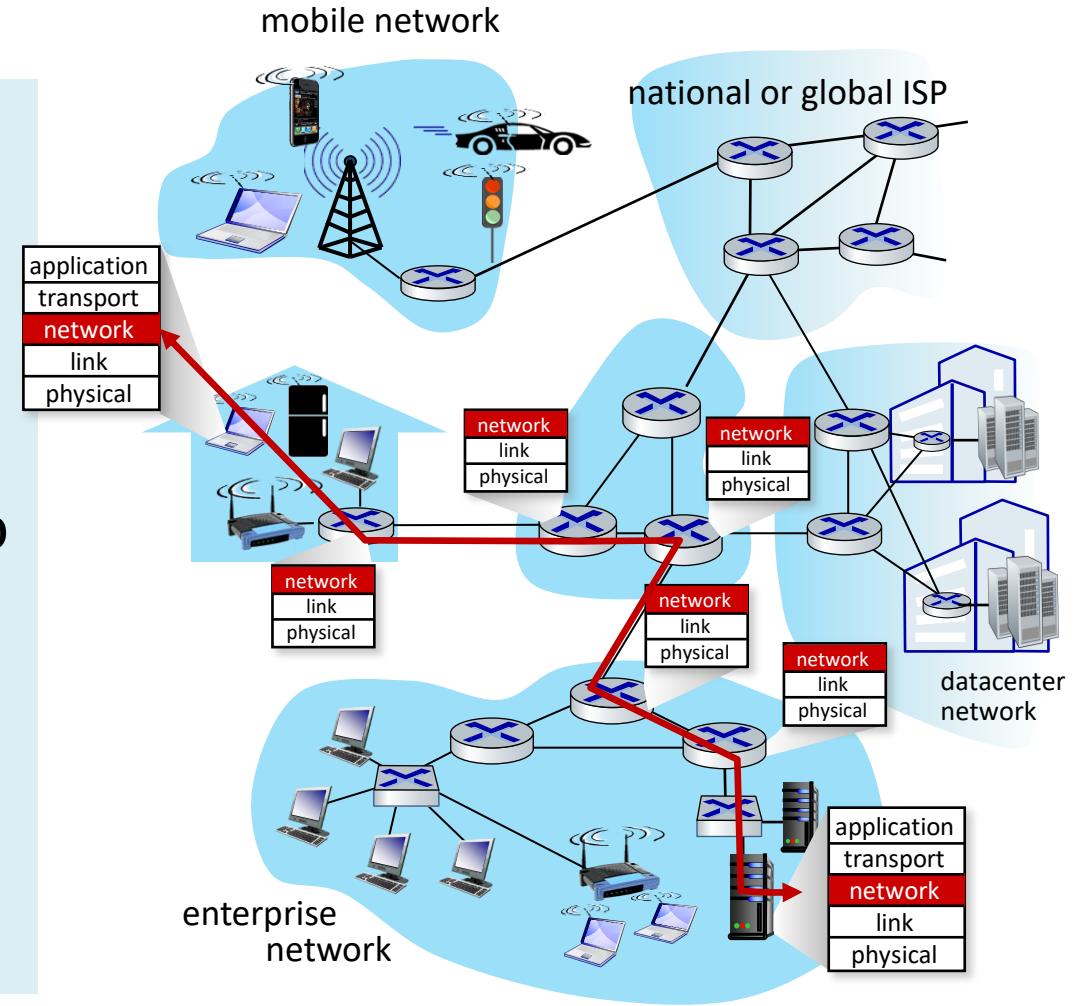
- Network layer: Data Plane
 - Forwarding vs routing
 - Network layer service models
 - Inside a router
 - The Internet Network layer: IP, Addressing & related
- Network layer: Control Plane
 - **Routing Protocols**
 - Link State
 - Distance Vector
 - Intra-ISP routing: OSPF
 - Routing among ISPs: BGP
 - Internet Control Message Protocol



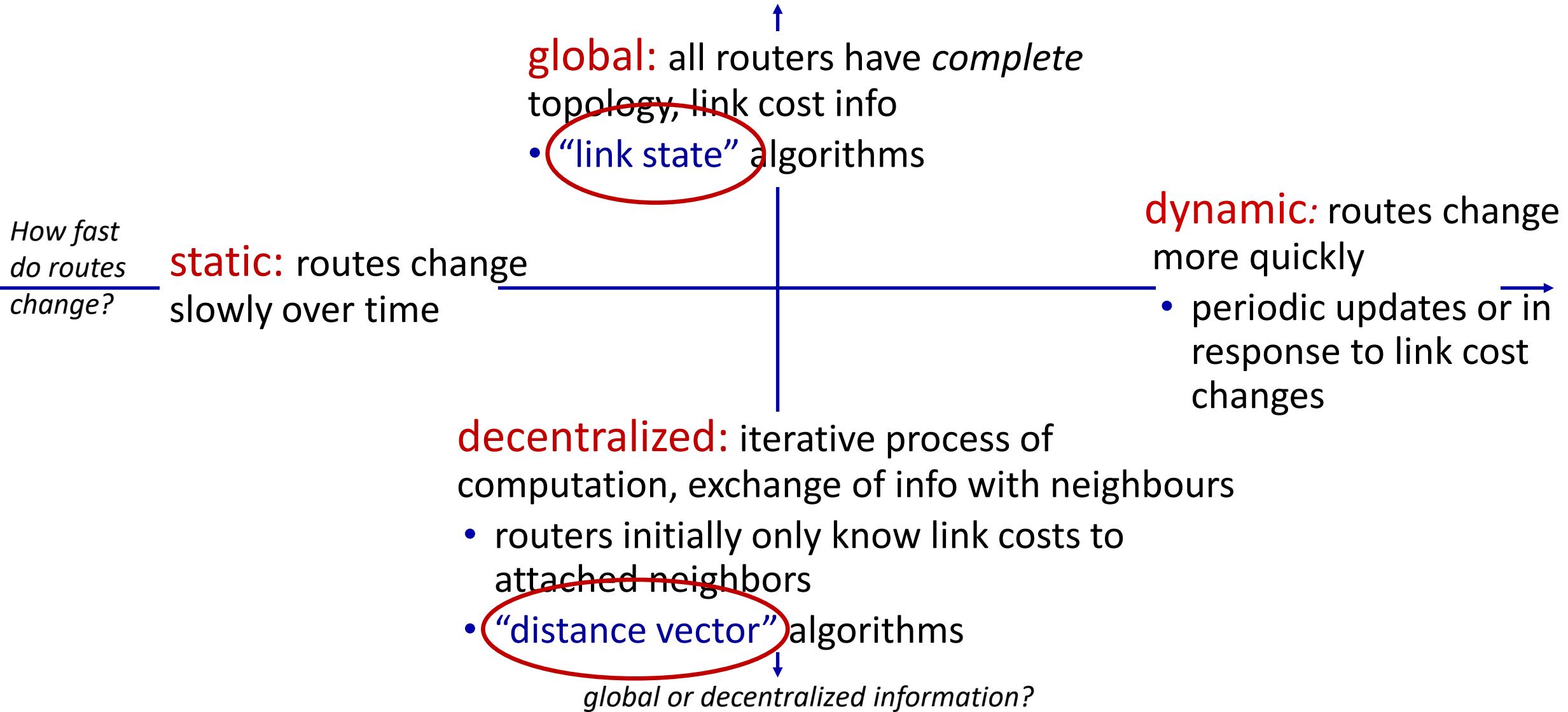
Routing protocols

Routing protocol goal: determine “good” paths (equivalently, routes), from sending hosts to receiving host, through network of routers

- **path:** sequence of routers packets traverse from given initial source host to final destination host
- **“good”:** least “cost”, “fastest”, “least congested”
- routing: a “top-10” networking challenge!



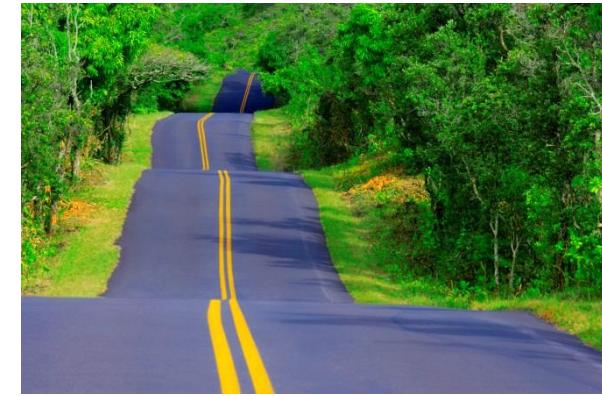
Routing algorithm classification



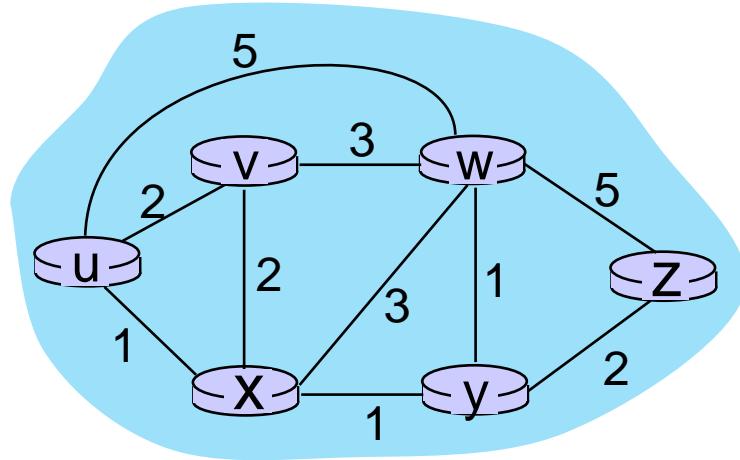
Roadmap Network Layer



- Network layer: Data Plane
 - Forwarding vs routing
 - Network layer service models
 - Inside a router
 - The Internet Network layer: IP, Addressing & related
- Network layer: Control Plane
 - Routing Protocols
 - Link State
 - Distance Vector
 - Intra-ISP routing: OSPF
 - Routing among ISPs: BGP
 - Internet Control Message Protocol



Graph abstraction: link costs



graph: $G = (N, E)$

N : set of routers = { u, v, w, x, y, z }

E : set of links = { $(u, v), (u, x), (v, x), (v, w), (w, x), (x, w), (x, y), (w, y), (w, z), (y, z)$ }

$c_{a,b}$: cost of *direct* link connecting a and b

e.g., $c_{w,z} = 5, c_{u,z} = \infty$

cost defined by network operator:
could always be 1, or inversely related
to bandwidth, or inversely related to
congestion

Dijkstra's link-state routing algorithm

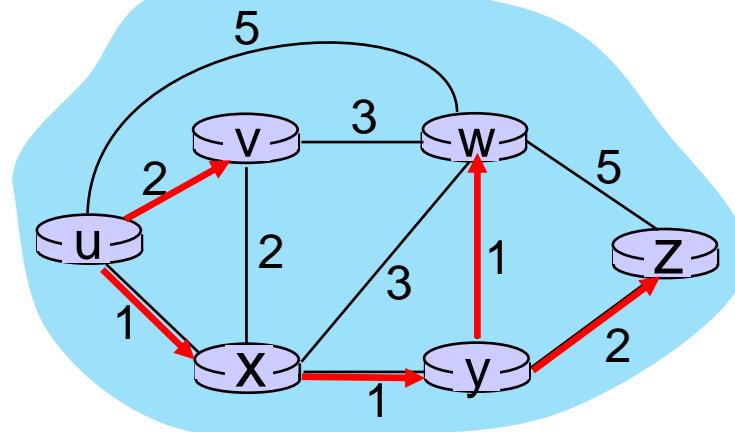
- **centralized:** network topology, link costs known to *all* nodes
 - accomplished via “link state broadcast”
 - all nodes have same info
- computes least cost paths from one node (“source”) to all other nodes
 - gives *forwarding table* for that node
- **iterative:** after k iterations, know least cost path to k destinations

notation

- $c_{x,y}$: direct link cost from node x to y ; $= \infty$ if not direct neighbors
- $D(v)$: *current estimate* of cost of least-cost-path from source to destination v
- $p(v)$: predecessor node along path from source to v
- N' : set of nodes whose least-cost-path *definitively* known

Dijkstra's algorithm: an example

Step	N'	$D(v), p(v)$	$D(w), p(w)$	$D(x), p(x)$	$D(y), p(y)$	$D(z), p(z)$
0	u	2, u	5, u	1, u	∞	∞
1	u, x	2, u	4, x	2, x	∞	∞
2	u, x, y	2, u	3, y	∞	4, y	∞
3	u, x, y, v	∞	3, y	∞	4, y	∞
4	u, x, y, v, w	∞	∞	∞	4, y	∞
5	u, x, y, v, w, z	∞	∞	∞	∞	∞

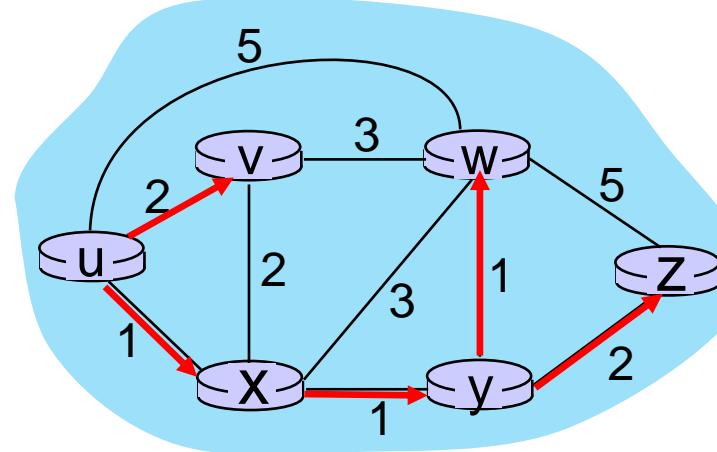


Initialization (step 0): For all a : if a adjacent to then $D(a) = c_{u,a}$

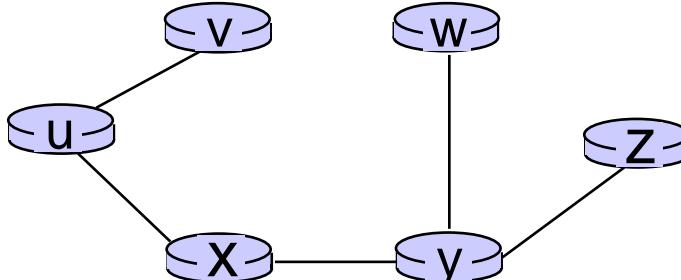
find a not in N' such that $D(a)$ is a minimum
 add a to N'
 update $D(b)$ for all b adjacent to a and not in N' :

$$D(b) = \min (D(b), D(a) + c_{a,b})$$

Dijkstra's algorithm: an example



resulting least-cost-path tree from u:



resulting forwarding table in u:

destination	outgoing link
v	(u,v)
x	(u,x)
y	(u,x)
w	(u,x)
x	(u,x)

route from u to v directly

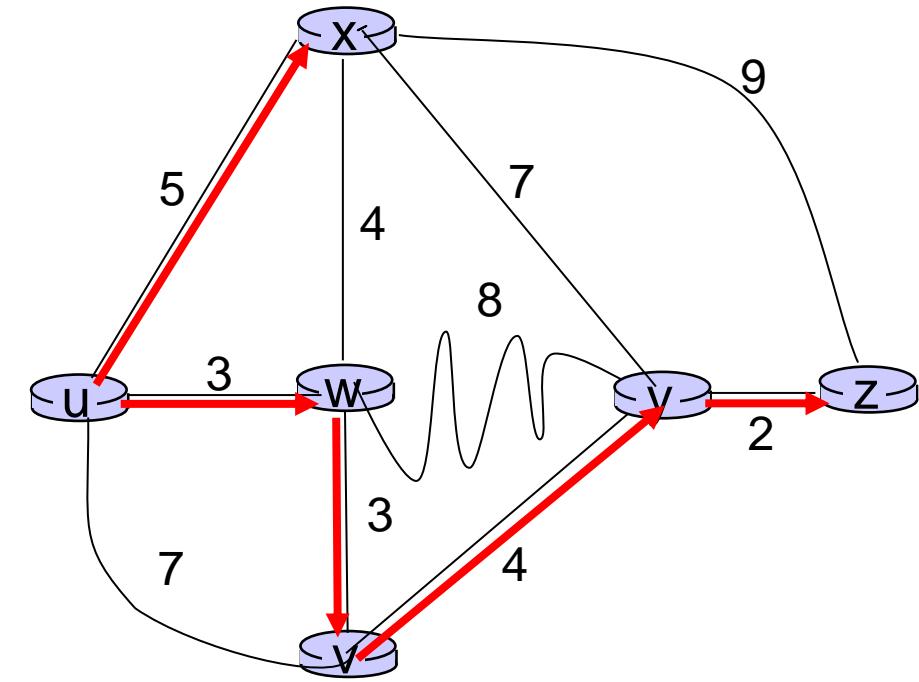
route from u to all other destinations via x

Dijkstra's link-state routing algorithm

```
1 Initialization:
2    $N' = \{u\}$                                 /* compute least cost path from u to all other nodes */
3   for all nodes v
4     if v adjacent to u                      /* u initially knows direct-path-cost only to direct neighbors */
5       then  $D(v) = c_{u,v}$                   /* but may not be minimum cost! */
6     else  $D(v) = \infty$ 
7
8 Loop
9   find w not in  $N'$  such that  $D(w)$  is a minimum
10  add w to  $N'$ 
11  update  $D(v)$  for all v adjacent to w and not in  $N'$ :
12     $D(v) = \min(D(v), D(w) + c_{w,v})$ 
13  /* new least-path-cost to v is either old least-cost-path to v or known
14  least-cost-path to w plus direct-cost from w to v */
15 until all nodes in  $N'$ 
```

Dijkstra's algorithm: another example

Step	N'	v	w	x	y	z
0	u	$D(v), p(v)$	$D(w), p(w)$	$D(x), p(x)$	$D(y), p(y)$	$D(z), p(z)$
1	uw	$7, u$	$3, u$	$5, u$	∞	∞
2	uwx	$6, w$	$5, u$	$11, w$	∞	
3	$uwxv$			$11, w$	$14, x$	
4	$uwxy$			$10, v$	$14, x$	
5	$uwxyz$				$12, y$	

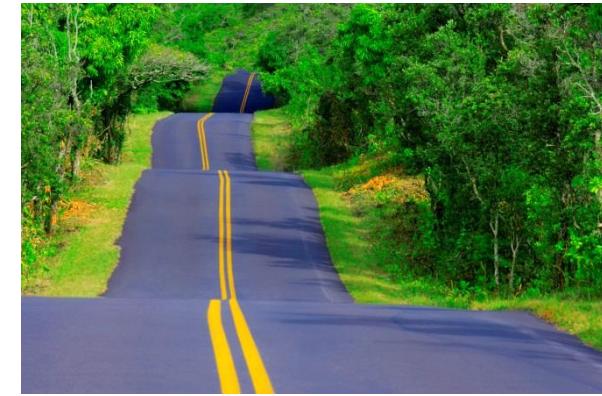


notes:

- construct least-cost-path tree by tracing predecessor nodes
- ties can exist (can be broken arbitrarily)

Roadmap Network Layer

- Network layer: Data Plane
 - Forwarding vs routing
 - Network layer service models
 - Inside a router
 - The Internet Network layer: IP, Addressing & related
- Network layer: Control Plane
 - **Routing Protocols**
 - Link State
 - Distance Vector
 - Intra-ISP routing: OSPF
 - Routing among ISPs: BGP
 - Internet Control Message Protocol



Distance vector algorithm

Based on *Bellman-Ford* (BF) equation (dynamic programming):

Bellman-Ford equation

Let $D_x(y)$: cost of least-cost path from x to y .

Then:

$$D_x(y) = \min_v \{ c_{x,v} + D_v(y) \}$$

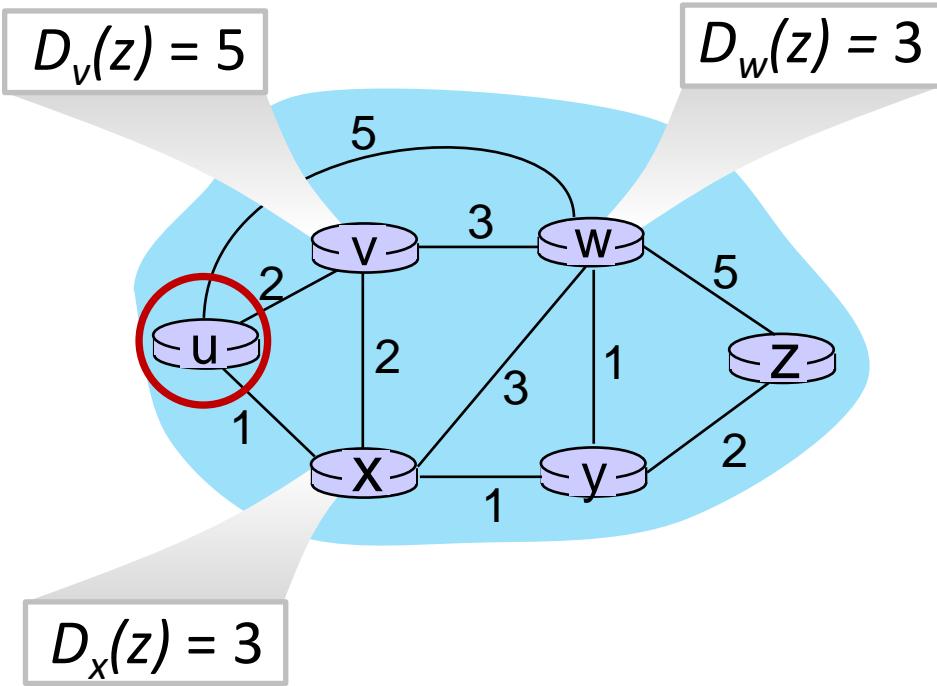
\min taken over all neighbors v of x

v 's estimated least-cost-path cost to y

direct cost of link from x to v

Bellman-Ford Example

Suppose that u 's neighboring nodes, x, v, w , know that for destination z :

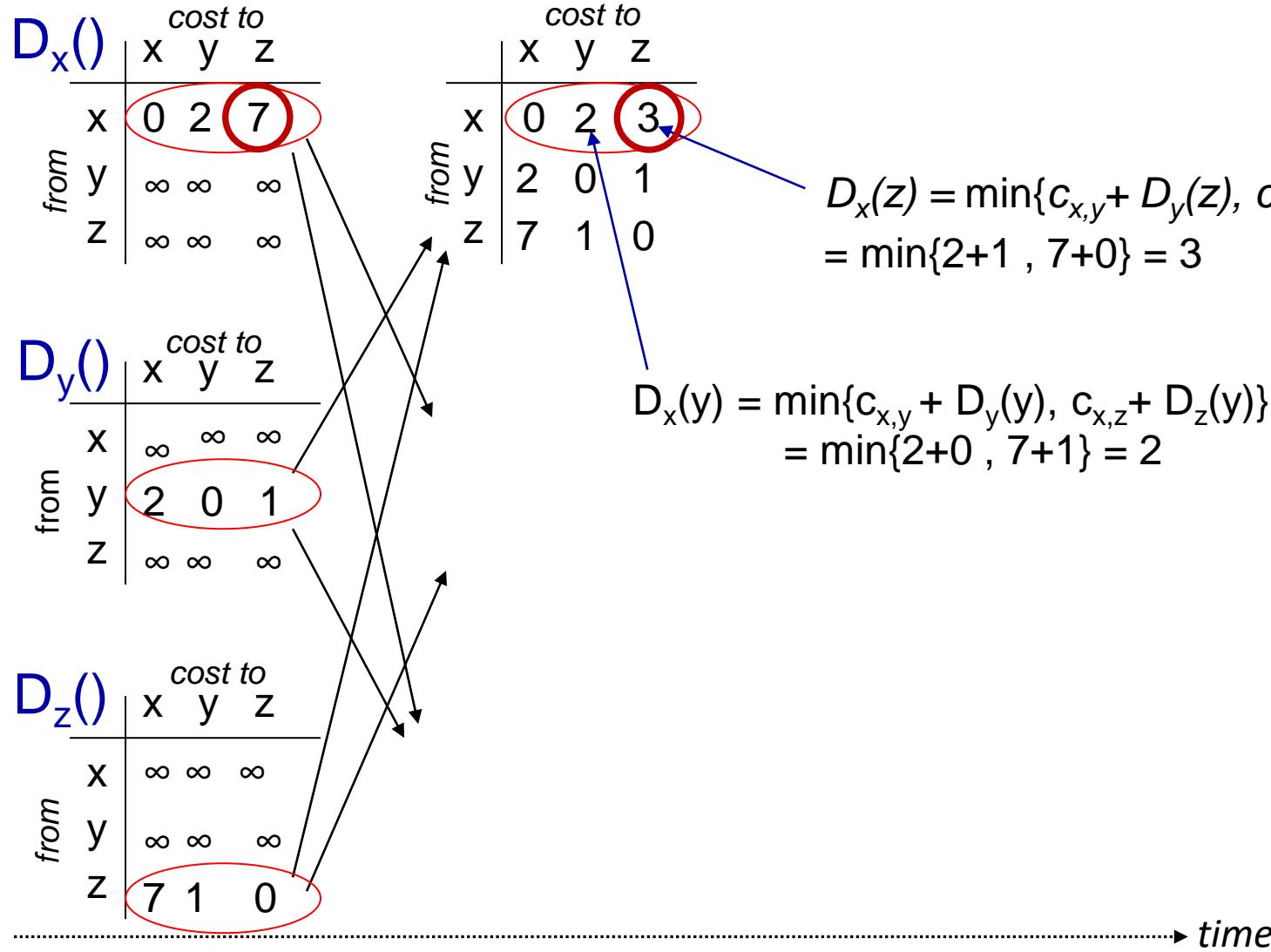


Bellman-Ford equation says:

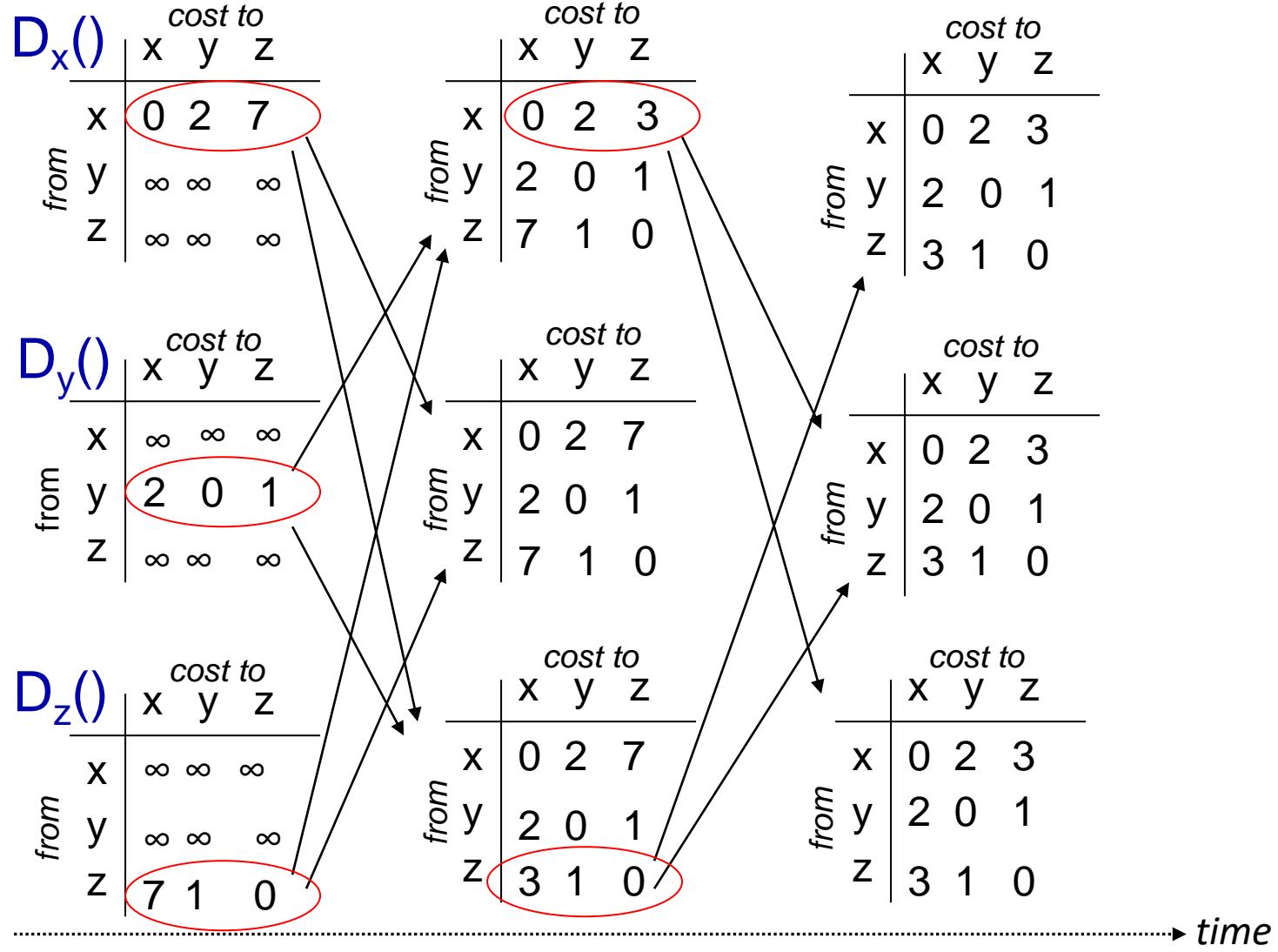
$$\begin{aligned} D_u(z) &= \min \{ c_{u,v} + D_v(z), \\ &\quad c_{u,x} + D_x(z), \\ &\quad c_{u,w} + D_w(z) \} \\ &= \min \{ 2 + 5, \\ &\quad 1 + 3, \\ &\quad 5 + 3 \} = 4 \end{aligned}$$

node achieving minimum (x) is next hop on estimated least-cost path to destination (z)

Distance vector: an example



Distance vector: an example



Distance vector algorithm

key idea:

- from time-to-time, each node sends its own distance vector estimate to neighbors
- when x receives new DV estimate from any neighbor, it updates its own DV using B-F equation:

$$D_x(y) \leftarrow \min_v \{c_{x,v} + D_v(y)\} \text{ for each node } y \in N$$

- under minor, natural conditions, the estimate $D_x(y)$ converge to the actual least cost $d_x(y)$

Distance vector algorithm

each node:

-
- ```
graph TD; A[wait for (change in local link cost or msg from neighbor)] --> B[recompute DV estimates using DV received from neighbor]; B --> C;if DV to any destination has changed, notify neighbors
```
- wait* for (change in local link cost or msg from neighbor)
  - recompute* DV estimates using DV received from neighbor
  - if DV to any destination has changed, *notify* neighbors

**iterative, asynchronous:** each local iteration caused by:

- local link cost change
- DV update message from neighbor

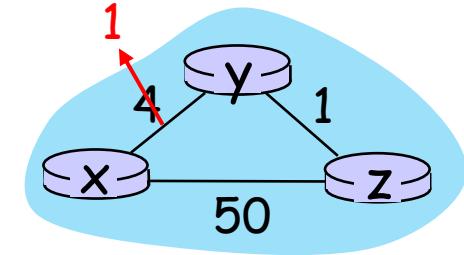
**distributed, self-stopping:** each node notifies neighbors *only* when its DV changes

- neighbors then notify their neighbors – *only if necessary*
- no notification received, no actions taken!

# Distance vector: link cost changes

## link cost changes:

- node detects local link cost change
- updates routing info, recalculates local DV
- if DV changes, notify neighbors



$t_0$ : y detects link-cost change, updates its DV, informs its neighbors.

“good news travels fast”

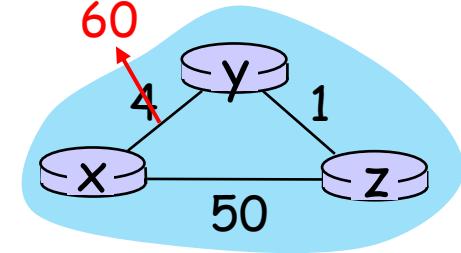
$t_1$ : z receives update from y, updates its table, computes new least cost to x , sends its neighbors its DV.

$t_2$ : y receives z's update, updates its distance table. y's least costs do *not* change, so y does *not* send a message to z.

# Distance vector: link cost changes

## link cost changes:

- node detects local link cost change
- “bad news travels slow” – count-to-infinity problem:
  - y sees direct link to x has new cost 60, but z has said it has a path at cost of 5. So y computes “my new cost to x will be 6, via z); notifies z of new cost of 6 to x.
  - z learns that path to x via y has new cost 6, so z computes “my new cost to x will be 7 via y), notifies y of new cost of 7 to x.
  - y learns that path to x via z has new cost 7, so y computes “my new cost to x will be 8 via y), notifies z of new cost of 8 to x.
  - z learns that path to x via y has new cost 8, so z computes “my new cost to x will be 9 via y), notifies y of new cost of 9 to x.
  - ...
- see text for solutions. *Distributed algorithms are tricky!*



# Comparison of Link State and Distance Vectors algorithm

---

## message complexity

LS:  $n$  routers,  $O(n^2)$  messages sent

DV: exchange between neighbors;  
convergence time varies

## speed of convergence

LS:  $O(n^2)$  algorithm,  $O(n^2)$  messages  
• may have oscillations

DV: convergence time varies  
• may have routing loops  
• count-to-infinity problem

## robustness: what happens if router malfunctions, or is compromised?

### LS:

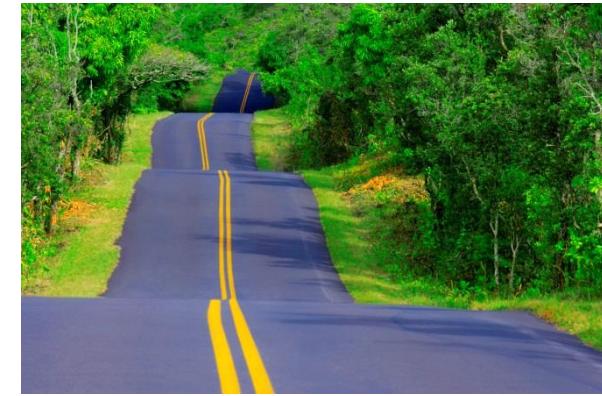
- router can advertise incorrect *link* cost
- each router computes only its *own* table

### DV:

- DV router can advertise incorrect *path* cost (“I have a *really* low cost path to everywhere”): black-holing
- each router’s table used by others: error propagate thru network

# Roadmap Network Layer

- Network layer: Data Plane
  - Forwarding vs routing
  - Network layer service models
  - Inside a router
  - The Internet Network layer: IP, Addressing & related
- Network layer: Control Plane
  - Routing Protocols
    - Link State
    - Distance Vector
  - Intra-ISP routing: OSPF
  - Routing among ISPs: BGP
  - Internet Control Message Protocol



# Making routing scalable

---

our routing study thus far - idealized

- all routers identical
- network “flat”

... not true in practice

**scale:** billions of destinations:

- can't store all destinations in routing tables!
- routing table exchange would swamp links!

**administrative autonomy:**

- Internet: a network of networks
- each network admin may want to control routing in its own network

# Internet approach to scalable routing

Routers are aggregated into regions known as “autonomous systems” (AS) (or “domains”)

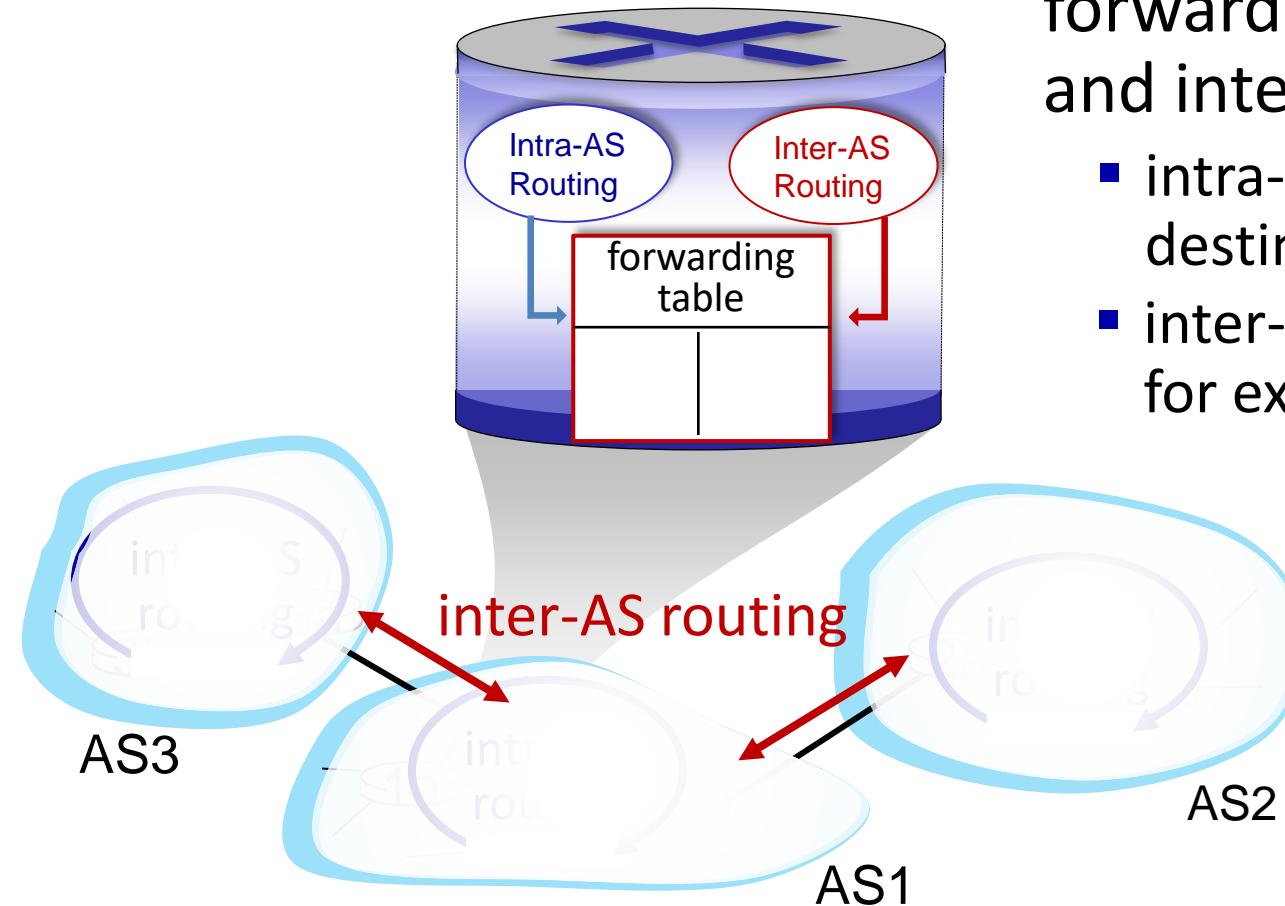
**intra-AS (aka “intra-domain”):**  
routing within same AS (“network”)

- all routers in AS must run same intra-domain protocol
- routers in different AS can run different intra-domain routing protocols
- **gateway router:** at “edge” of its own AS, has link(s) to router(s) in other AS'es

**inter-AS (aka “inter-domain”):**  
routing *among* AS'es

- gateways perform inter-domain routing (as well as intra-domain routing)

# Interconnected Autonomous Systems (AS)



forwarding table configured by intra-  
and inter-AS routing algorithms

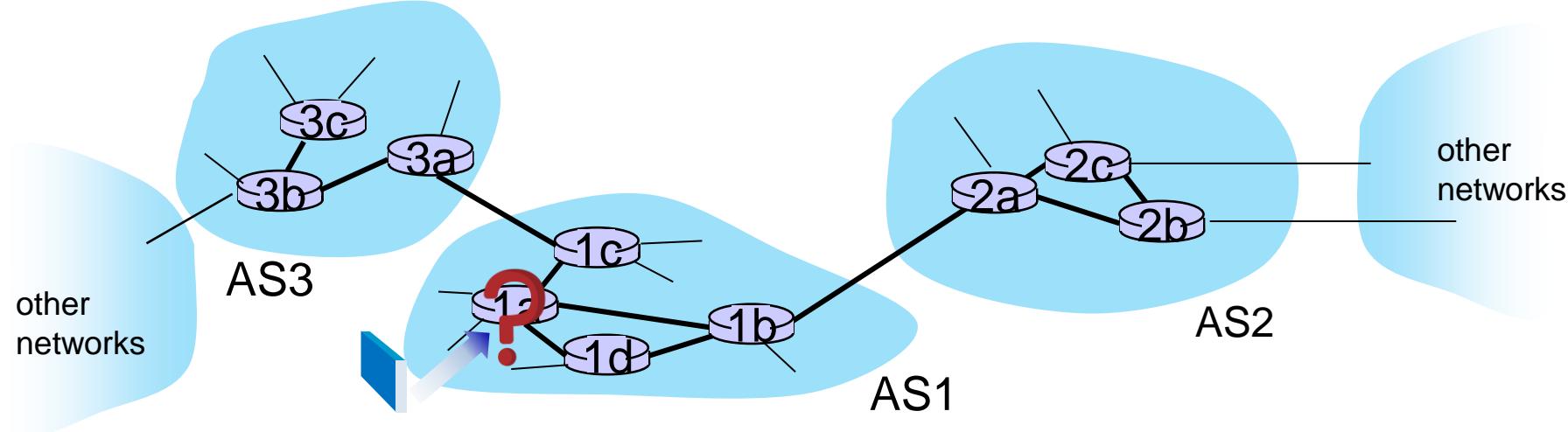
- intra-AS routing determine entries for destinations within AS
- inter-AS & intra-AS determine entries for external destinations

# Inter-AS routing: a role in intradomain forwarding

- suppose router in AS1 receives a datagram destined outside of AS1:
  - router should forward packet to gateway router in AS1, but which one?

**AS1 inter-domain routing must:**

1. learn which destinations reachable through AS2, which through AS3
2. propagate this reachability info to all routers in AS1



# Intra-AS routing: routing within an AS

Some examples of intra-AS routing protocols known also as **Interior gateway protocols**:

- **RIP: Routing Information Protocol [RFC 1723]**
  - classic DV: DVs exchanged every 30 secs
  - no longer widely used
- **EIGRP: Enhanced Interior Gateway Routing Protocol**
  - DV based
  - formerly Cisco-proprietary for decades (became open in 2013 [RFC 7868])
- **OSPF: Open Shortest Path First [RFC 2328]**
  - link-state routing <- Dijkstra ☺
  - IS-IS (Intermediate System to Intermediate System) protocol (ISO standard, not RFC standard) essentially same as OSPF

# OSPF (Open Shortest Path First) routing

---

- “open”: publicly available
- classic link-state
  - each router floods OSPF link-state advertisements (directly over IP rather than using TCP/UDP) to all other routers in entire AS
  - multiple link costs metrics possible: bandwidth, delay
  - **each router has full topology**, uses Dijkstra’s algorithm to compute forwarding table
- *security*: all OSPF messages authenticated (to prevent malicious intrusion)

# Hierarchical OSPF

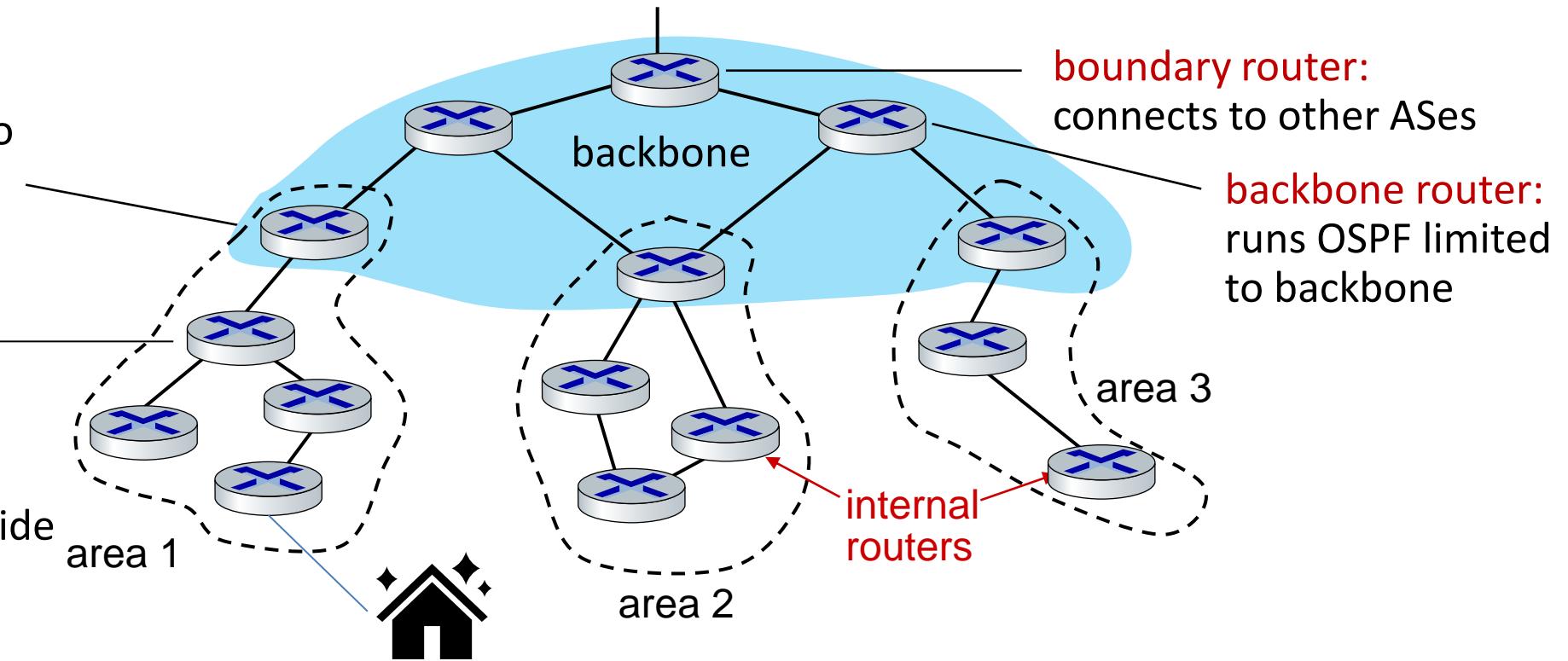
- two-level hierarchy: local area, backbone.
  - link-state advertisements flooded only in area, or backbone
  - each node has detailed area topology; only knows direction to reach other destinations

area border routers:

“summarize” distances to destinations in own area, advertise in backbone

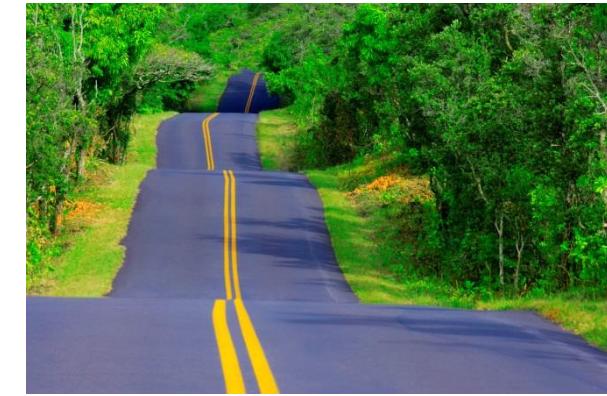
local routers:

- flood LS in area only
- compute routing within area
- forward packets to outside via area border router



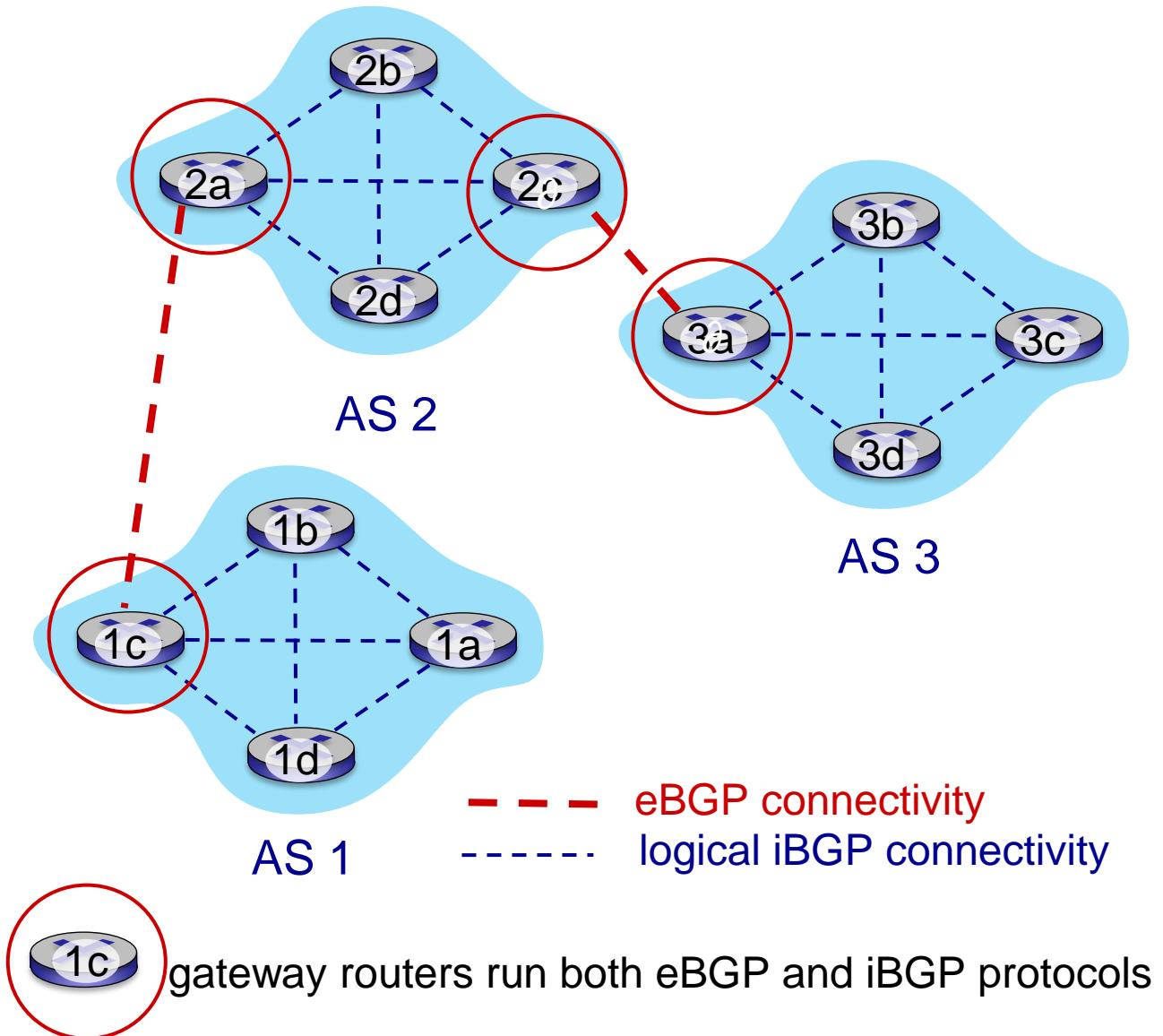
# Roadmap Network Layer

- Network layer: Data Plane
  - Forwarding vs routing
  - Network layer service models
  - Inside a router
  - The Internet Network layer: IP, Addressing & related
- Network layer: Control Plane
  - Routing Protocols
    - Link State
    - Distance Vector
  - Intra-ISP routing: OSPF
  - Routing among ISPs: BGP
  - Internet Control Message Protocol



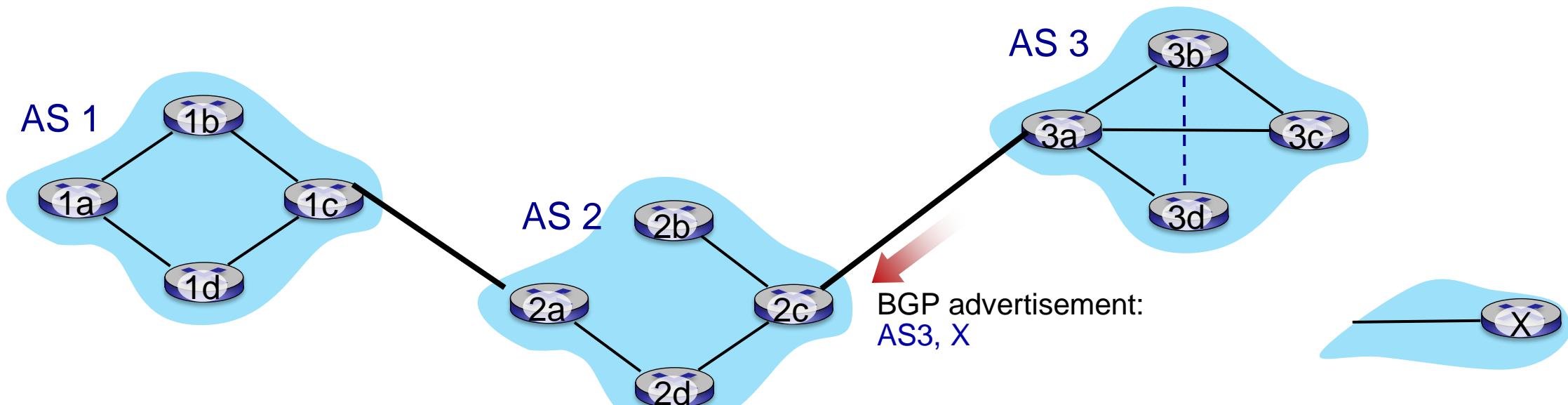
# Internet Inter-AS Routing: BGP

- BGP (Border Gateway Protocol):  
*the de facto* inter-domain routing protocol
  - “glue that holds the Internet together”
- BGP provides each Autonomous Systems (AS) means to:
  - **eBGP:** obtain subnet reachability information from neighboring ASes
  - **iBGP:** propagate reachability information to all AS-internal routers.
  - Determine “good” routes to other networks based on reachability information and *policy*



# BGP basics

- **BGP session:** two BGP routers (“peers”) exchange BGP messages over semi-permanent TCP connection:
  - advertising *paths* to different destination network prefixes (BGP is a “path vector” protocol)
- when AS3 gateway 3a advertises **path AS3,X** to AS2 gateway 2c:
  - AS3 *promises* to AS2 it will forward datagrams towards X

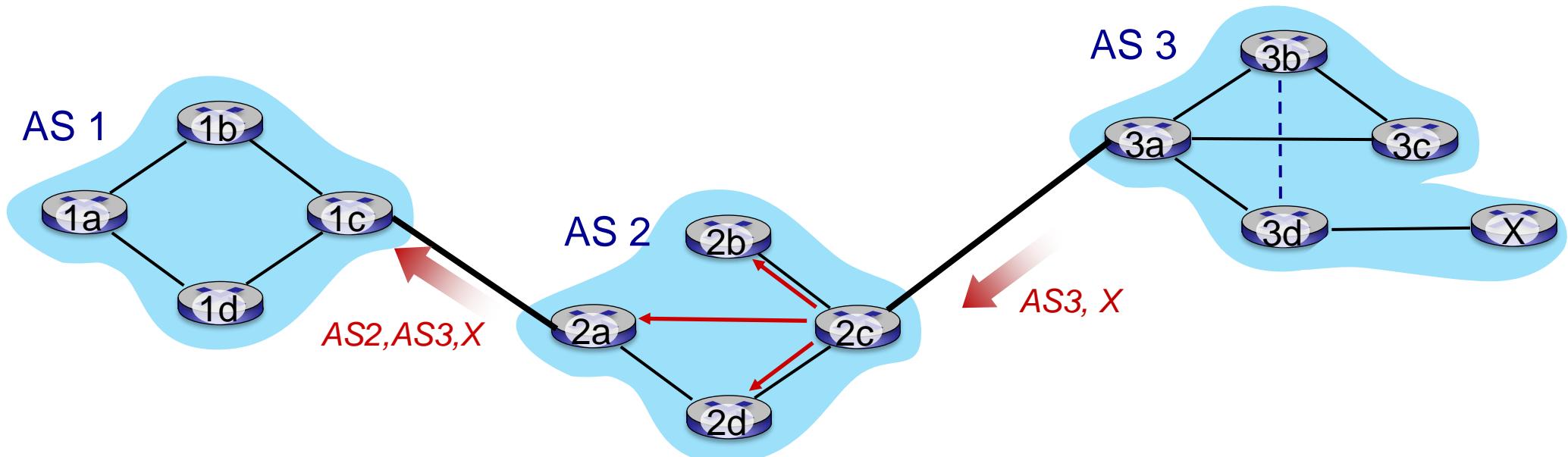


# Path attributes and BGP routes

---

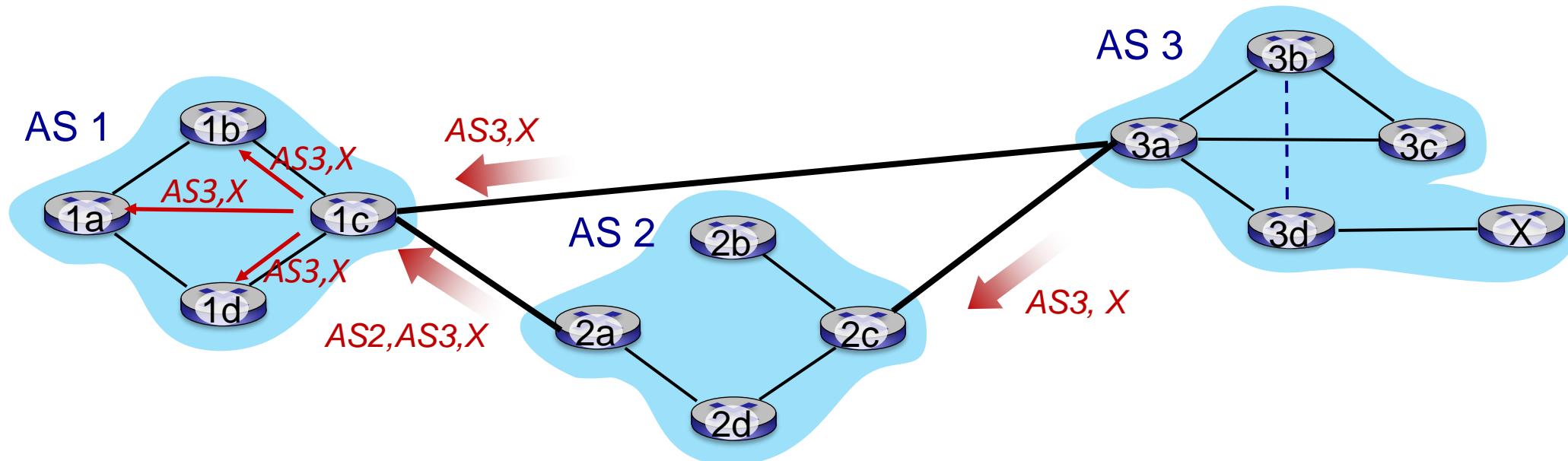
- BGP advertised route: prefix + attributes
  - prefix: destination being advertised
  - two important attributes:
    - AS-PATH: list of ASes through which prefix advertisement has passed
    - NEXT-HOP: indicates specific internal-AS router to next-hop AS
- policy-based routing:
  - gateway receiving route advertisement uses *import policy* to accept/decline path (e.g., never route through AS Y).
  - AS policy also determines whether to *advertise* path to other other neighboring ASes

# BGP path advertisement



- AS2 router 2c receives path advertisement **AS3,X** (via eBGP) from AS3 router 3a
- based on AS2 policy, AS2 router 2c accepts path AS3,X, propagates (via iBGP) to all AS2 routers
- based on AS2 policy, AS2 router 2a advertises (via eBGP) path **AS2, AS3, X** to AS1 router 1c

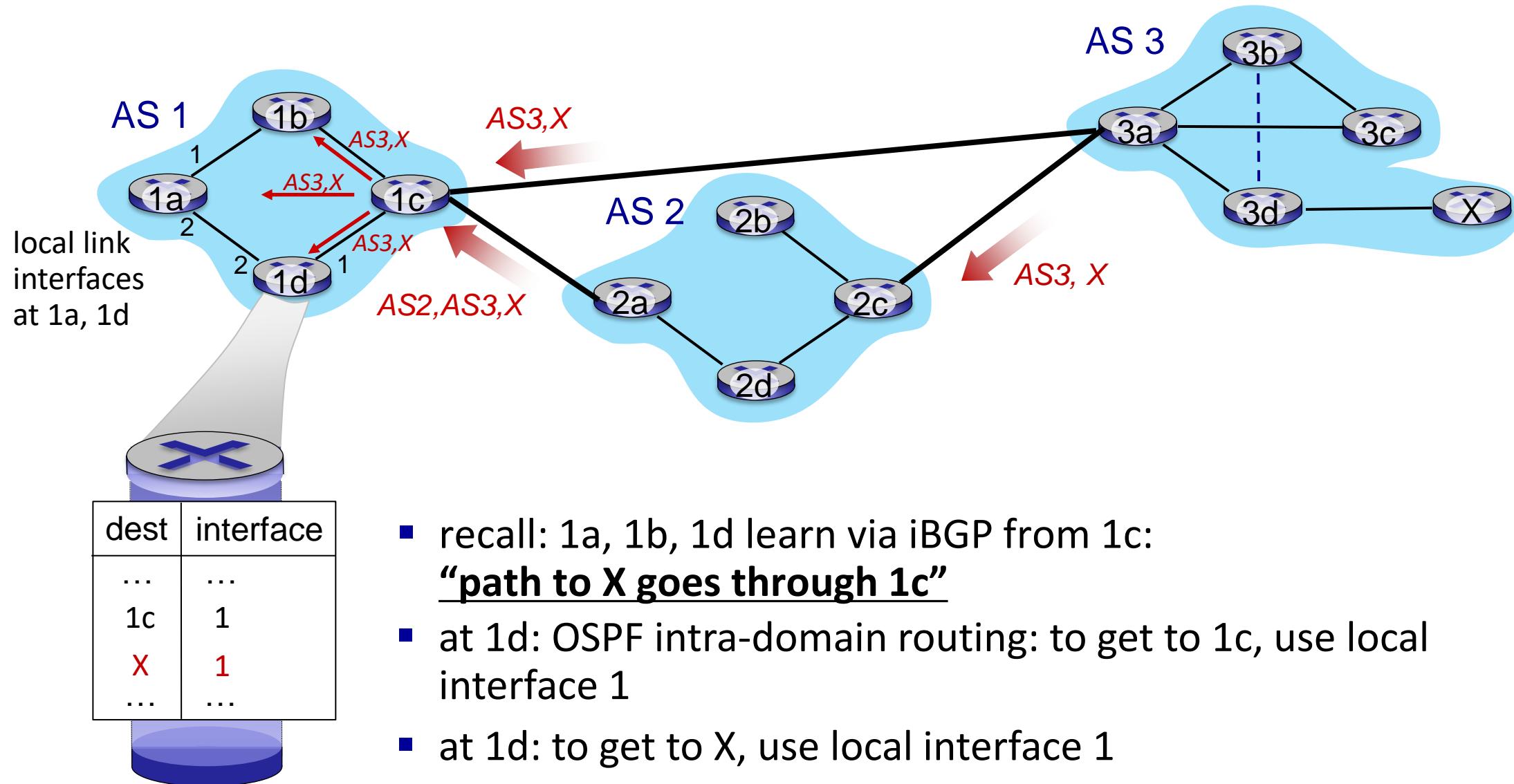
# BGP path advertisement (with multiple possible paths)



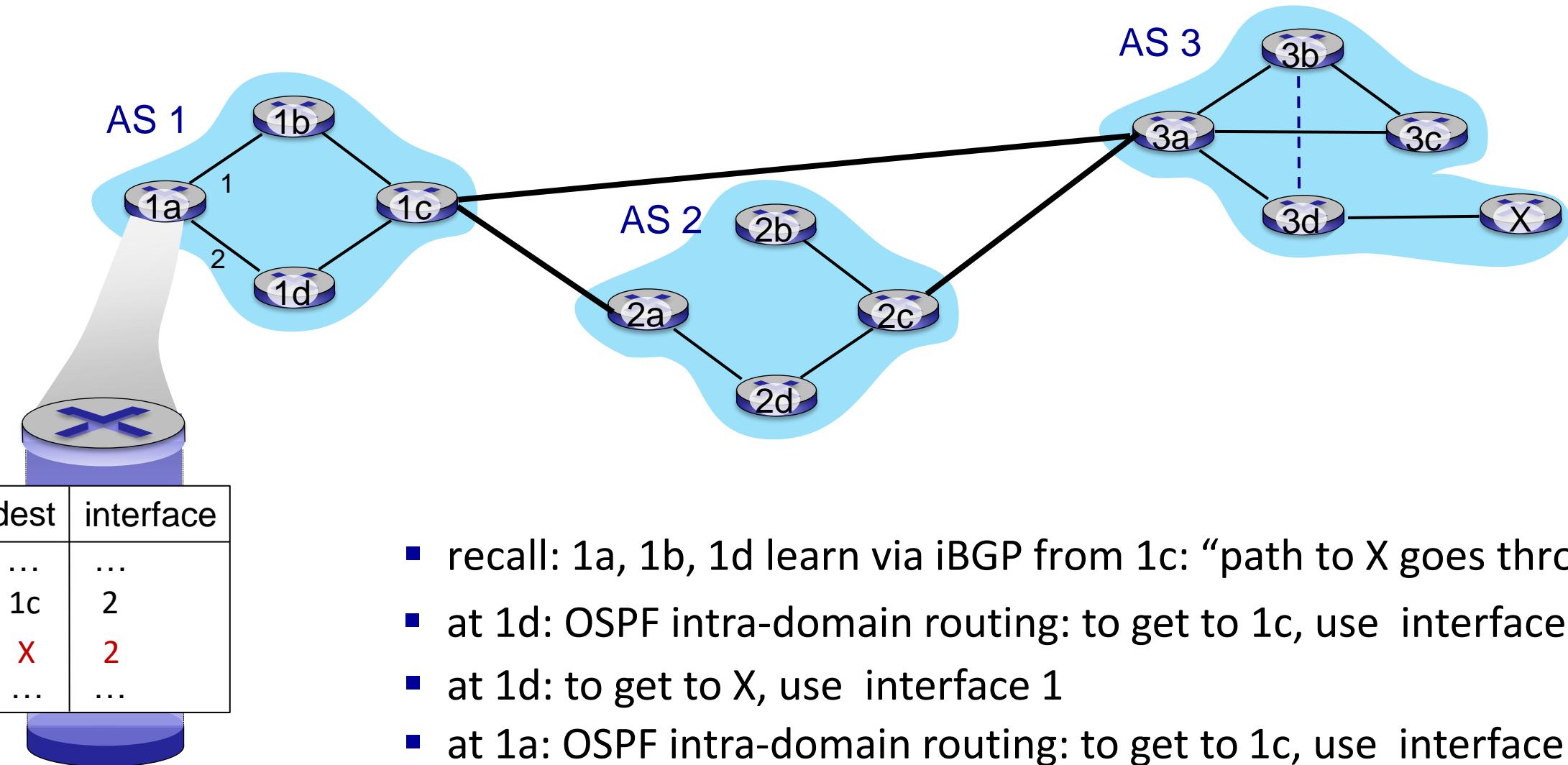
gateway router may learn about **multiple** paths to destination:

- AS1 gateway router 1c learns path ***AS2,AS3,X*** from 2a via eBGP
- AS1 gateway router 1c learns path ***AS3,X*** from 3a via eBGP
- based on *policy*, AS1 gateway router 1c chooses path ***AS3,X*** and advertises path within AS1 via iBGP

# BGP path advertisement



# BGP path advertisement

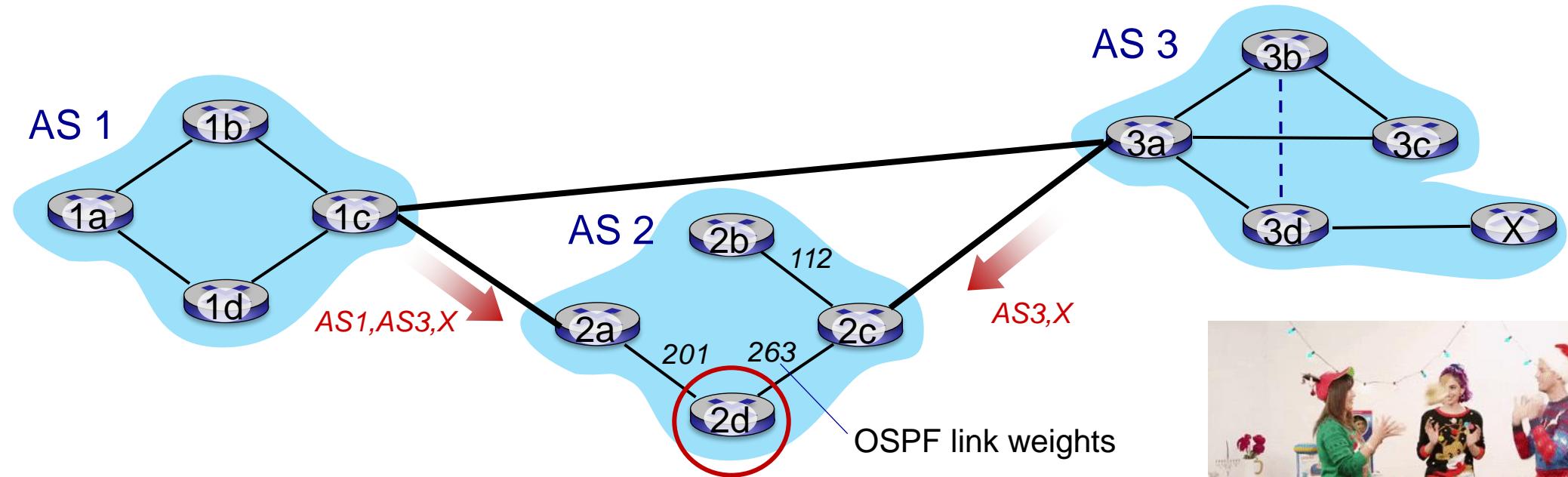


# BGP Route Selection Criteria

---

- Router may learn about more than one route to destination AS, selects route based on:
  1. local preference value attribute: policy decision
  2. shortest AS-PATH
  3. closest NEXT-HOP router: hot potato routing
  4. additional criteria

# Hot potato routing



- 2d learns (via iBGP) it can route to X via 2a or 2c
- **hot potato routing:** choose local gateway that has least *intra-domain cost* (e.g., 2d chooses 2a, even though more AS hops to X):
  - Get packets out of the AS as quickly as possible. Don't worry about **inter-domain cost!**

# Why different Intra-, Inter-AS routing?

---

policy:

- inter-AS: admin wants control over how its traffic routed, who routes through its network
- intra-AS: single admin, so policy less of an issue

scale:

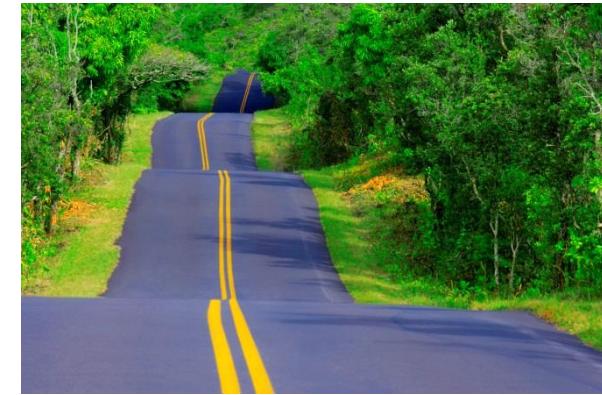
- hierarchical routing saves table size, reduced update traffic

performance:

- intra-AS: can focus on performance
- inter-AS: policy dominates over performance

# Roadmap Network Layer

- - Network layer: Data Plane
    - Forwarding vs routing
    - Network layer service models
    - Inside a router
    - The Internet Network layer: IP, Addressing & related
  - Network layer: Control Plane
    - Routing Protocols
      - Link State
      - Distance Vector
    - Intra-ISP routing: OSPF
    - Routing among ISPs: BGP
    - Internet Control Message Protocol

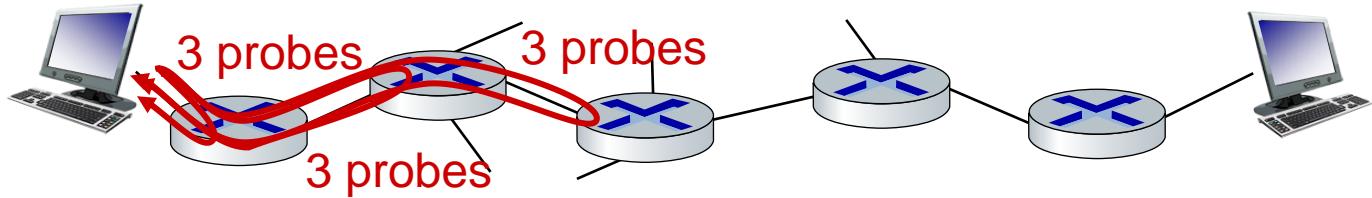


# ICMP: Internet Control Message Protocol

- Used by hosts and routers to communicate network-level information
  - error reporting: unreachable host, network, port, protocol
  - echo request/reply (used by ping)
- Network-layer “above” IP:
  - ICMP messages carried in IP datagrams
- *ICMP message*: type, code plus first 8 bytes of IP datagram (that e.g., caused an error)

| Type | Code | description                                   |
|------|------|-----------------------------------------------|
| 0    | 0    | echo reply (ping)                             |
| 3    | 0    | dest. network unreachable                     |
| 3    | 1    | dest host unreachable                         |
| 3    | 2    | dest protocol unreachable                     |
| 3    | 3    | dest port unreachable                         |
| 3    | 6    | dest network unknown                          |
| 3    | 7    | dest host unknown                             |
| 4    | 0    | source quench (congestion control - not used) |
| 8    | 0    | echo request (ping)                           |
| 9    | 0    | route advertisement                           |
| 10   | 0    | router discovery                              |
| 11   | 0    | TTL expired                                   |
| 12   | 0    | bad IP header                                 |

# Traceroute and ICMP



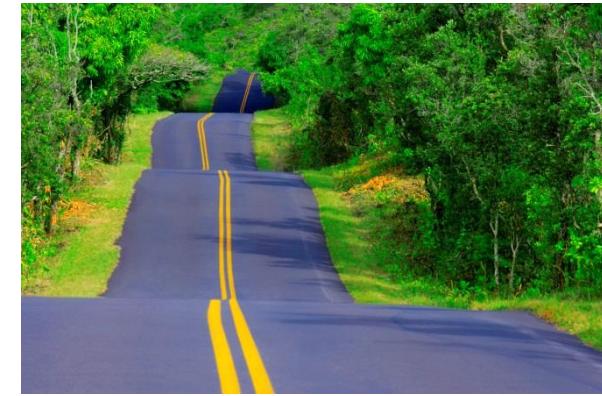
- source sends sets of UDP segments to destination
  - 1<sup>st</sup> set has TTL =1, 2<sup>nd</sup> set has TTL=2, etc.
- datagram in *n*th set arrives to *n*th router:
  - router discards datagram and sends source ICMP message (type 11, code 0)
  - ICMP message possibly includes name of router & IP address
- when ICMP message arrives at source: record RTTs

## stopping criteria:

- UDP segment eventually arrives at destination host
- destination returns ICMP “port unreachable” message (type 3, code 3)
- source stops

# Roadmap Network Layer

- Network layer: Data Plane
  - Forwarding vs routing
  - Network layer service models
  - Inside a router
  - The Internet Network layer: IP, Addressing & related
- Network layer: Control Plane
  - Routing Protocols
    - Link State
    - Distance Vector
  - Intra-ISP routing: OSPF
  - Routing among ISPs: BGP
  - Internet Control Message Protocol



# Layering: The OSI Reference Model

ISO (International Standards Organization) defined the OSI (Open Systems Interconnect) model to help vendors create interoperable network implementation

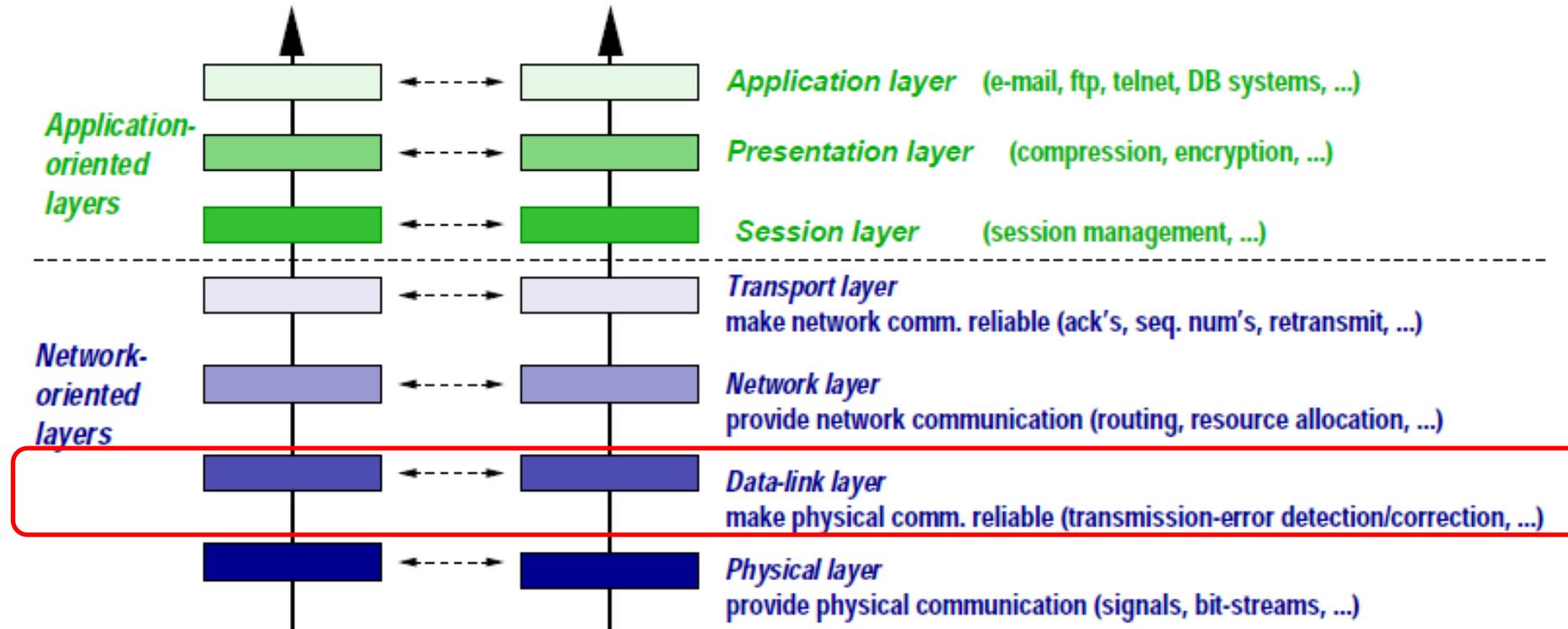


Fig. Steen, Sips : Computer and Network organization

``X dot'' series (X.25, X. 400, X.500) OSI model implementation (protocol stack)



# Course on Computer Communication and Networks

## Lecture 8 Chapter 6; Link Layer and LAN Part 1

EDA344/DIT423/LEU062

Lecturer: Hans-Martin Heyn

Based on the book Computer Networking: A Top Down Approach, Jim Kurose, Keith Ross, Addison-Wesley.

# Layering: The OSI Reference Model

ISO (International Standards Organization) defined the OSI (Open Systems Interconnect) model to help vendors create interoperable network implementation

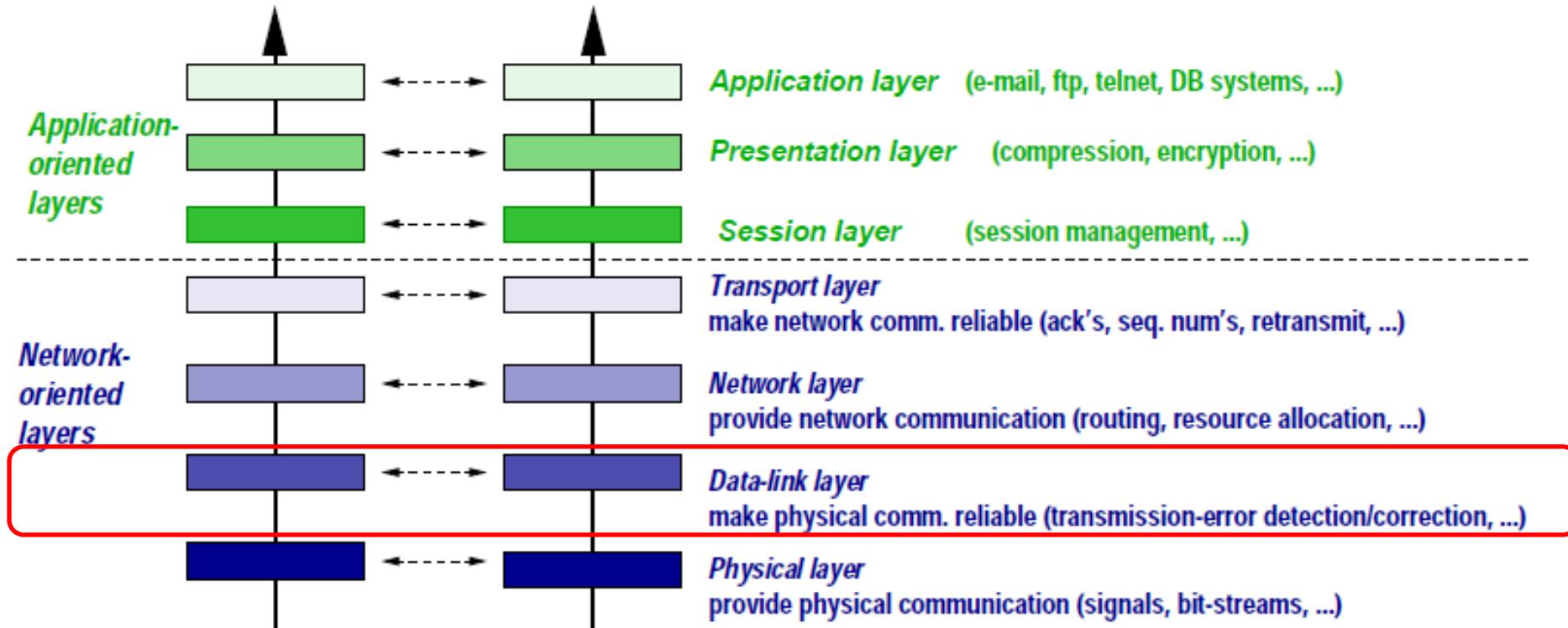
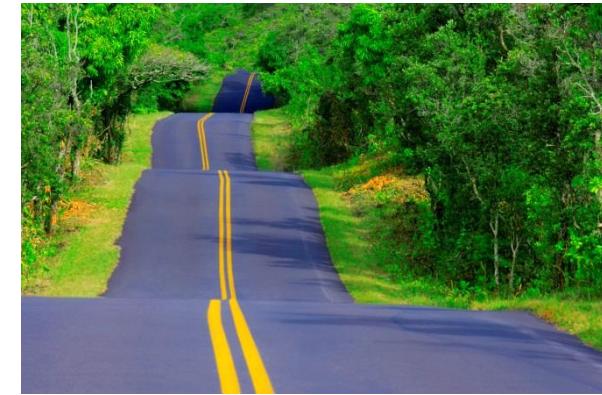


Fig. Steen, Sips : Computer and Network organization

``X dot'' series (X.25, X. 400, X.500) OSI model implementation (protocol stack)

# Roadmap Data-Link Layer

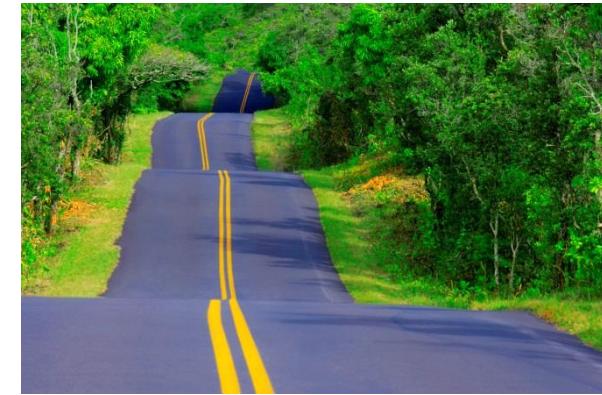
- Introduction to the link layer
- Error-Detection, Error-Correction
  - Parity Checks
  - Cyclic Redundancy Check (CRC)
- Multiple access protocols (MAC)
  - Channel Partitioning
  - Random Access Protocols
  - Taking-Turns Protocols
- Local Area Networks (LANs)
- Virtual Networks (VLANs)
- Data Centre Networking



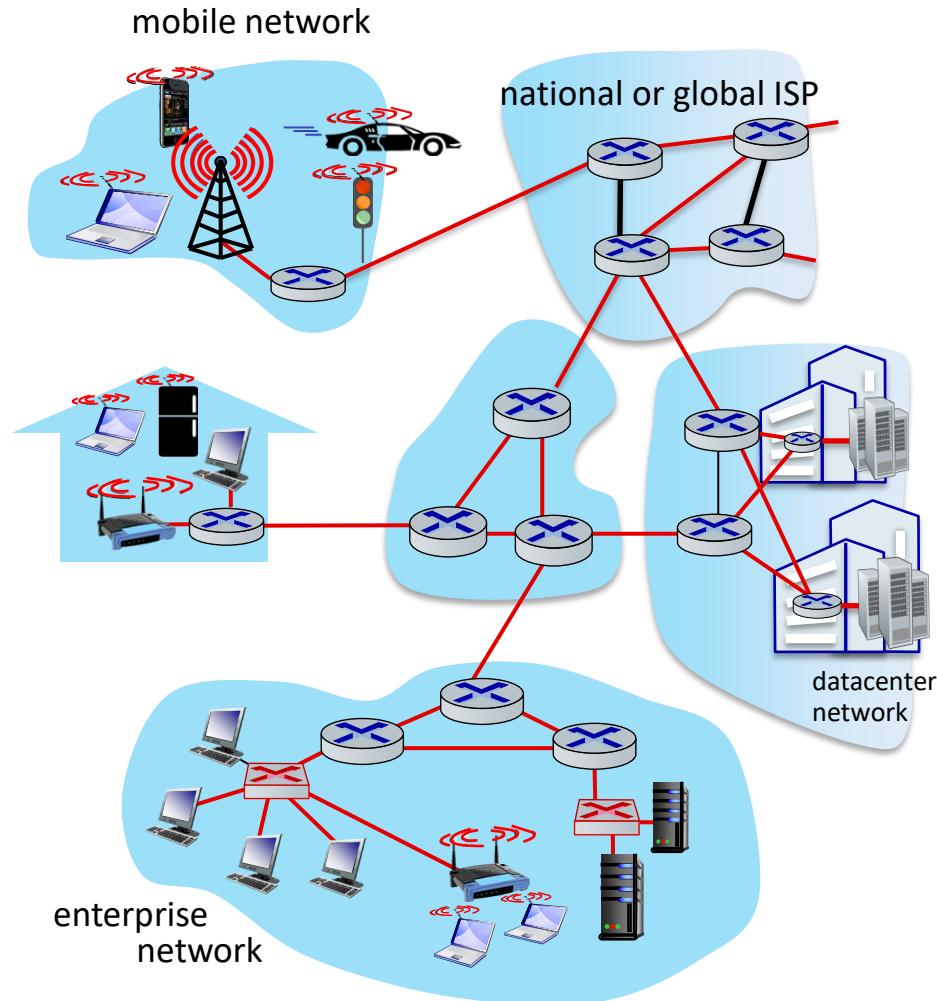
# Roadmap Data-Link Layer



- Introduction to the link layer
- Error-Detection, Error-Correction
  - Parity Checks
  - Cyclic Redundancy Check (CRC)
- Multiple access protocols (MAC)
  - Channel Partitioning
  - Random Access Protocols
  - Taking-Turns Protocols
- Local Area Networks (LANs)
- Virtual Networks (VLANs)
- Data Centre Networking



# Introduction: Some Terminology



## Terminology:

- hosts and routers: nodes
- communication channels that connect adjacent nodes along communication path: links
  - wired
  - wireless
  - LANs
- Layer-2 (data-link) packet: *frame*, encapsulates datagram

*(data-)link layer* has responsibility of transferring datagram from one node to **physically adjacent** node over a link

# Link layer: Travelling datagrams

---

- Datagram transferred by different link protocols over different links:
  - e.g., WiFi on first link, Ethernet on next link
- Each link protocol provides different services
  - e.g., **may or may not** provide reliable data transfer over link



## Transportation analogy:

- Trip from Chalmers to Lausanne
  - Taxi: Johanneberg to Landvetter
  - Plane: Landvetter to Geneva
  - Train: Geneva to Lausanne
- Tourist = **datagram**
- Transport segment = **communication link**
- Transportation mode = **link-layer protocol**
- Travel agent = **routing algorithm**

# Link layer: Services

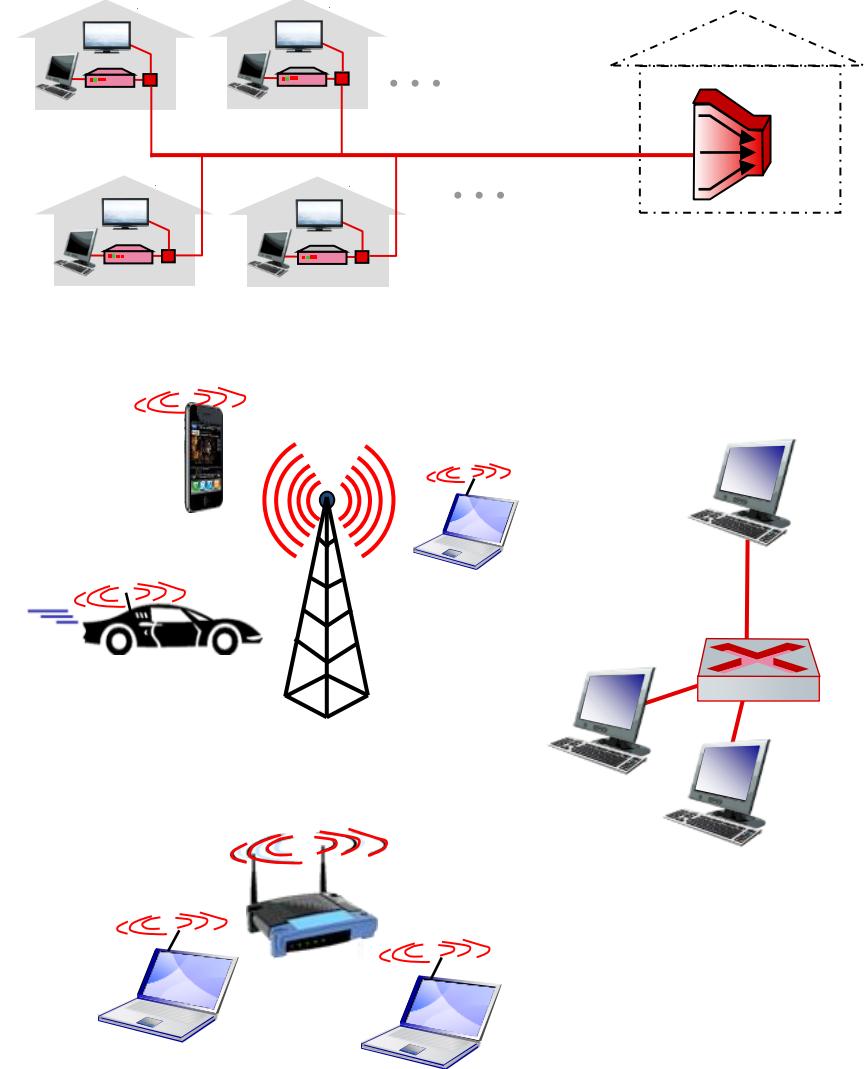
- **Framing, link access:**

- encapsulate datagram into frame, adding header, trailer
- channel access if shared medium
- “MAC” addresses in frame headers identify source, destination (different from IP address!)

- **Reliable delivery between adjacent nodes**

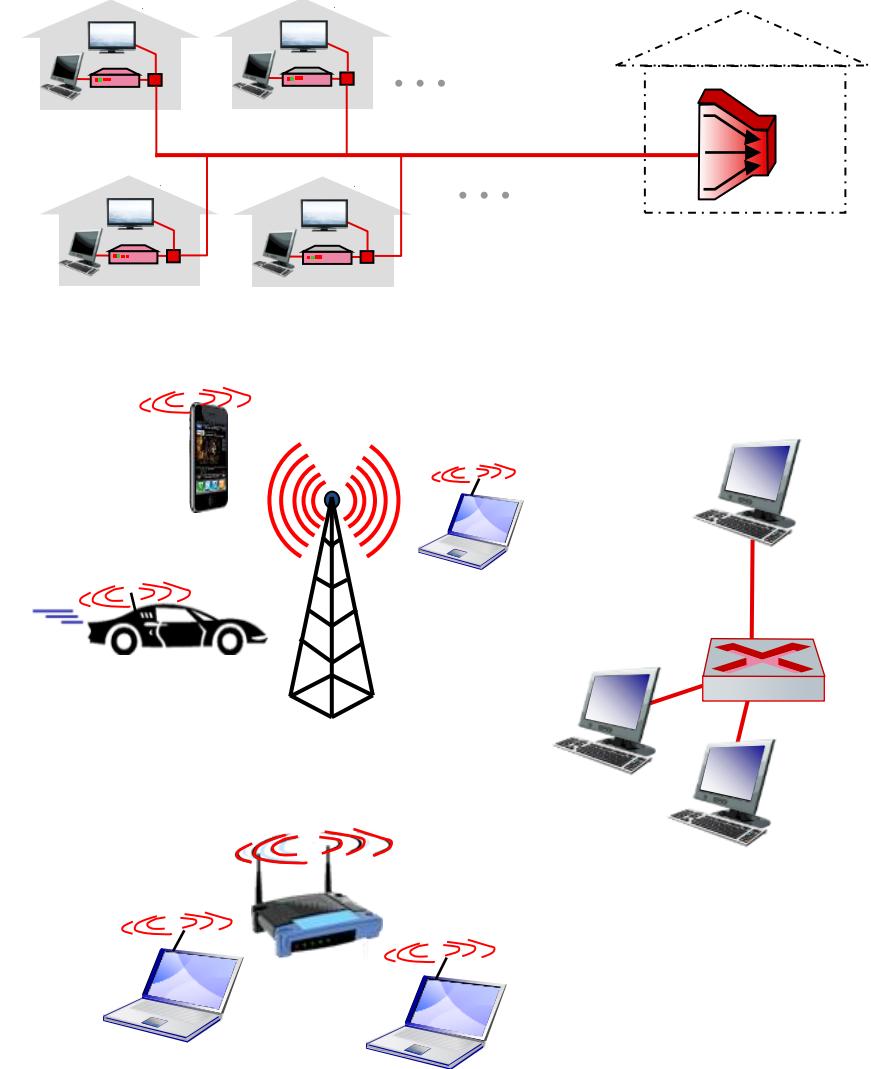
- we already know how to do this!
- seldom used on low bit-error links
- **wireless links: high error rates**

- Q: Why implement reliable data transfer on both link layer and end-end reliability (e.g., TCP on transport layer?)



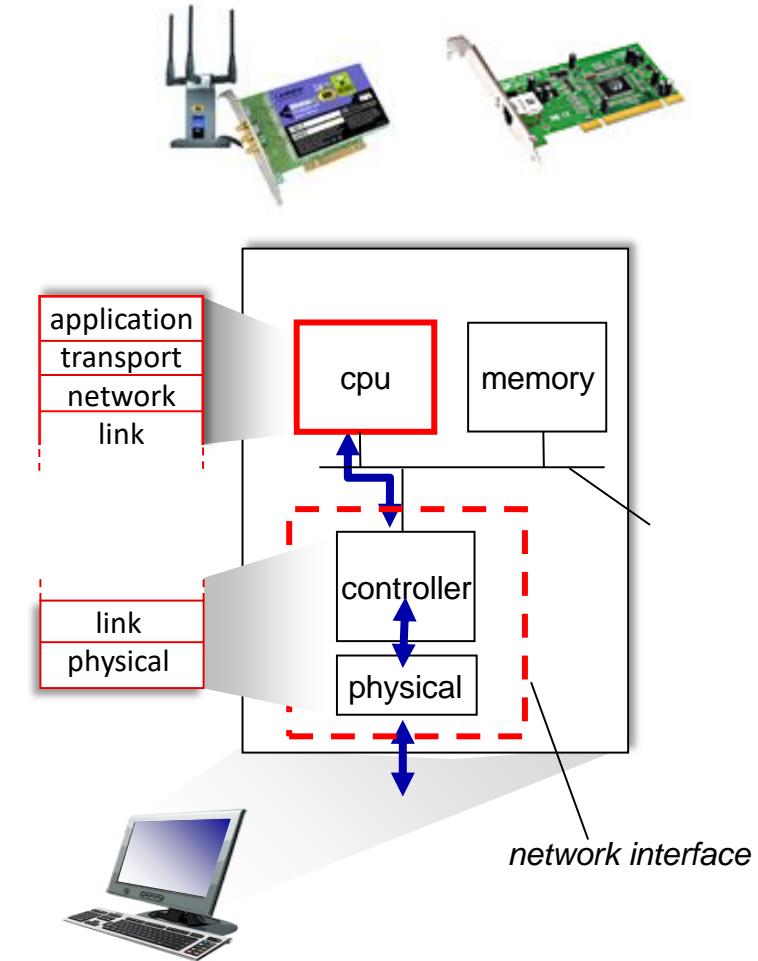
# Link layer: Services

- **Flow control:**
  - pacing between adjacent sending and receiving nodes
- **Error detection:**
  - errors caused by signal attenuation, noise.
  - receiver detects errors, signals retransmission, or drops frame
- **Error correction:**
  - receiver identifies *and corrects* bit error(s) without retransmission
- **Half-duplex and full-duplex:**
  - with half duplex, nodes at both ends of link can transmit, but not at same time

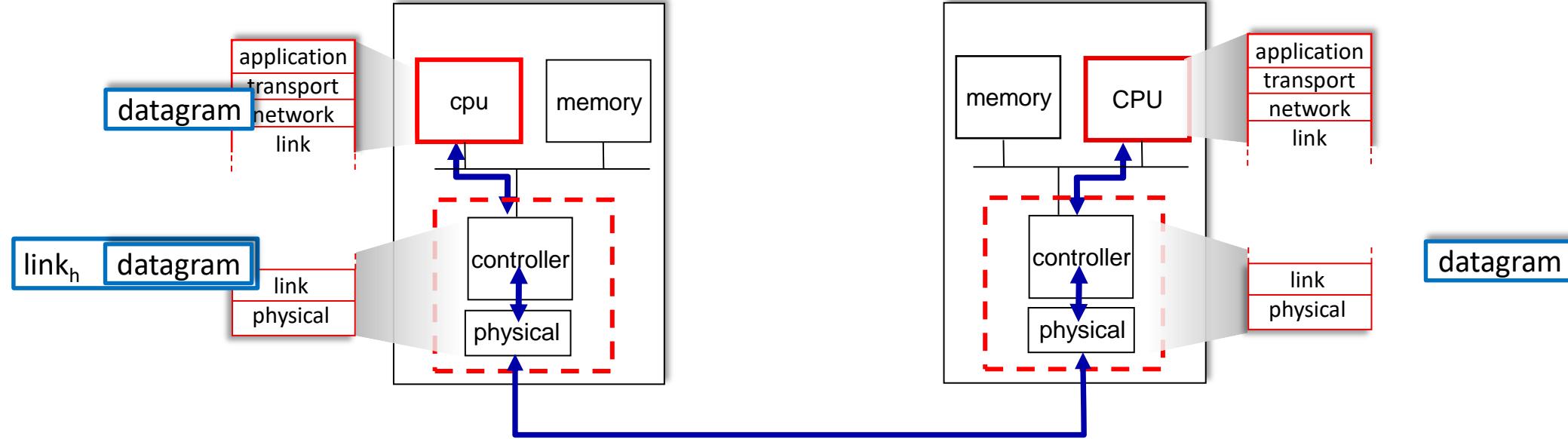


# Where is the link layer implemented?

- In each-and-every host
- Link layer implemented in *network interface card* (NIC) (or embedded on a chip)
  - Ethernet, WiFi card or chip
  - implements link, physical layer
- Attaches into host's system buses
- Combination of hardware, software, firmware



# Interfaces communicating



Sending side:

- encapsulates datagram in frame
- adds error checking bits, reliable data transfer, flow control, etc.

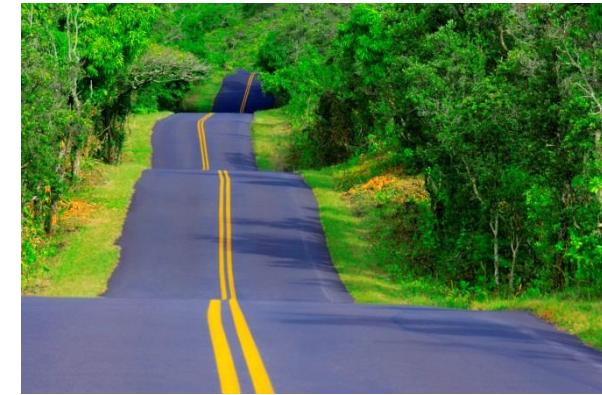
Receiving side:

- looks for errors, reliable data transfer, flow control, etc.
- extracts datagram, passes to upper layer at receiving side

# Roadmap Data-Link Layer



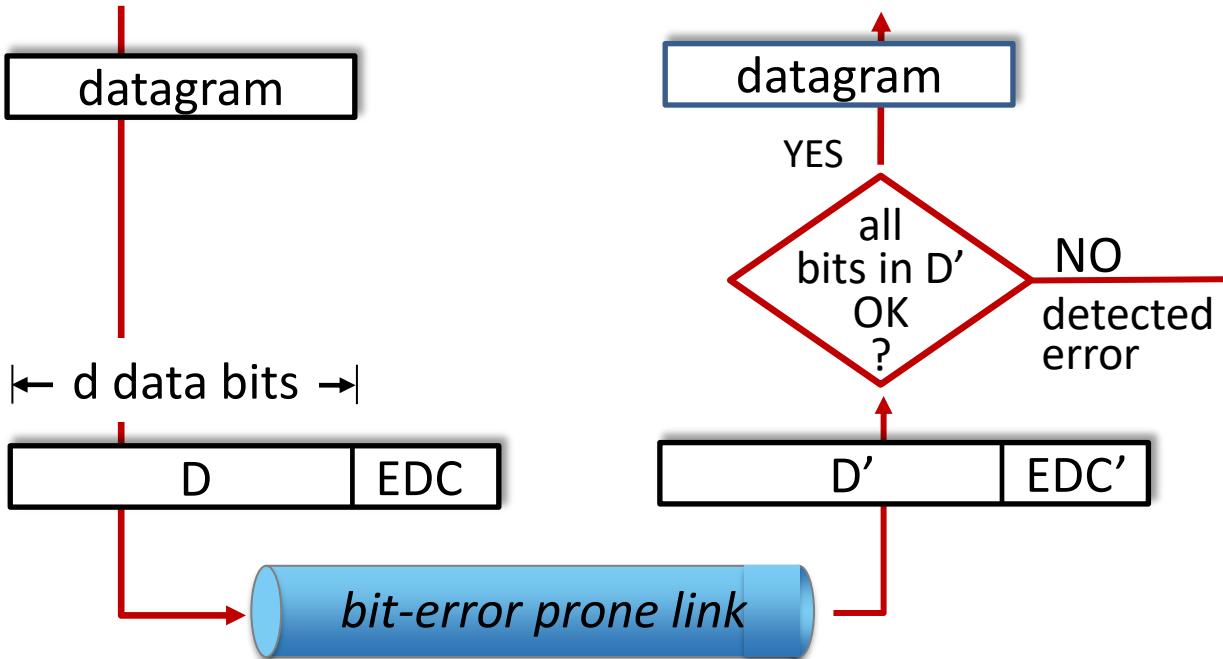
- Introduction to the link layer
- Error-Detection, Error-Correction
  - Parity Checks
  - Cyclic Redundancy Check (CRC)
- Multiple access protocols (MAC)
  - Channel Partitioning
  - Random Access Protocols
  - Taking-Turns Protocols
- Local Area Networks (LANs)
- Virtual Networks (VLANs)
- Data Centre Networking



# Error detection

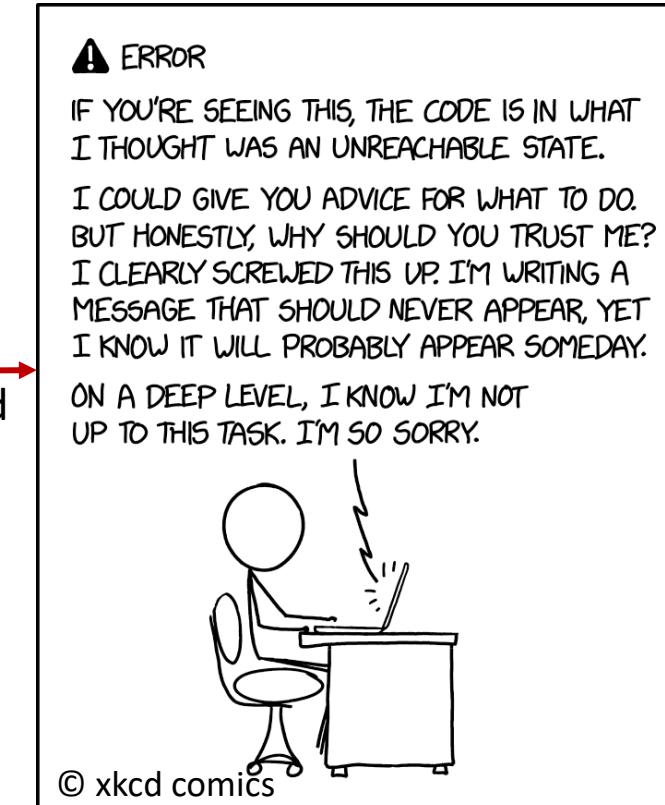
**EDC:** Error detection and correction bits (e.g., redundancy)

**D:** Data protected by error checking, may include header fields



Error detection not 100% reliable!

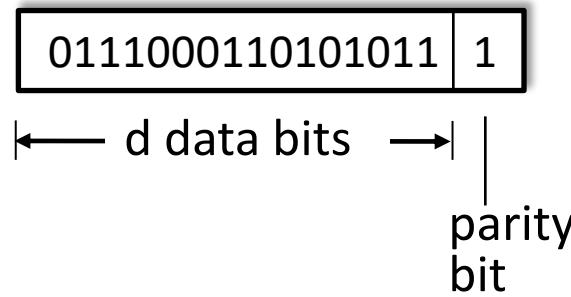
- protocol may miss some errors, but rarely
- larger EDC field yields better detection and correction



# Parity checking

## Single bit parity:

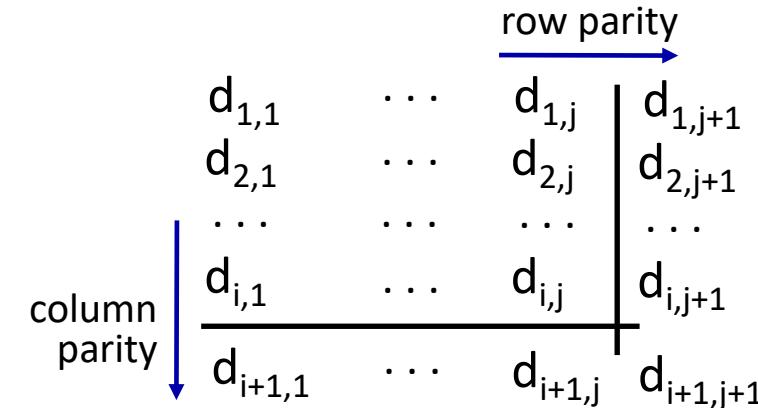
- detect single bit errors



Even parity: set parity bit so there is an even number of 1's

## Two-dimensional bit parity:

- detect *and correct* single bit errors



no errors: 
$$\begin{array}{c|c} 1 & 0 \\ 1 & 1 \\ 0 & 1 \\ \hline 0 & 0 \end{array} \quad \begin{array}{c|c} 1 & 1 \\ 0 & 1 \\ 1 & 0 \\ \hline 0 & 0 \end{array}$$

detected  
and  
correctable  
single-bit  
error:

A grid of binary digits with a red cross through the second column. A red arrow points from the text "detected and correctable single-bit error:" to the second column. Another red arrow points to the bottom-right corner of the grid with the label "parity error".

# Internet checksum

---

**Goal:** detect errors (*i.e.*, flipped bits) in transmitted segment

## Sender:

- treat contents of UDP segment (including UDP header fields and IP addresses) as sequence of 16-bit integers
- **checksum:** addition (one's complement sum) of segment content
- checksum value put into UDP checksum field

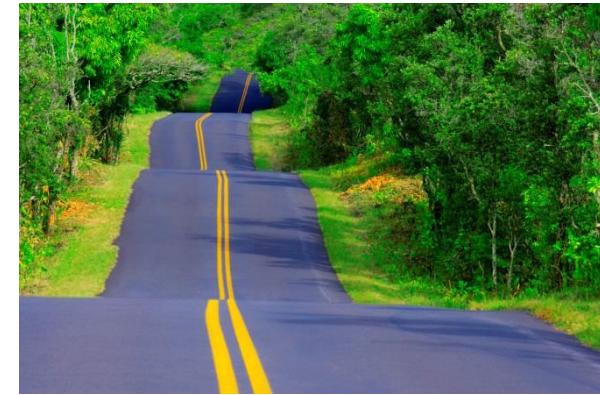
## Receiver:

- compute checksum of received segment
- check if computed checksum equals checksum field value:
  - not equal - error detected
  - equal - no error detected. **But maybe errors nonetheless?** More later ....

# Roadmap Data-Link Layer

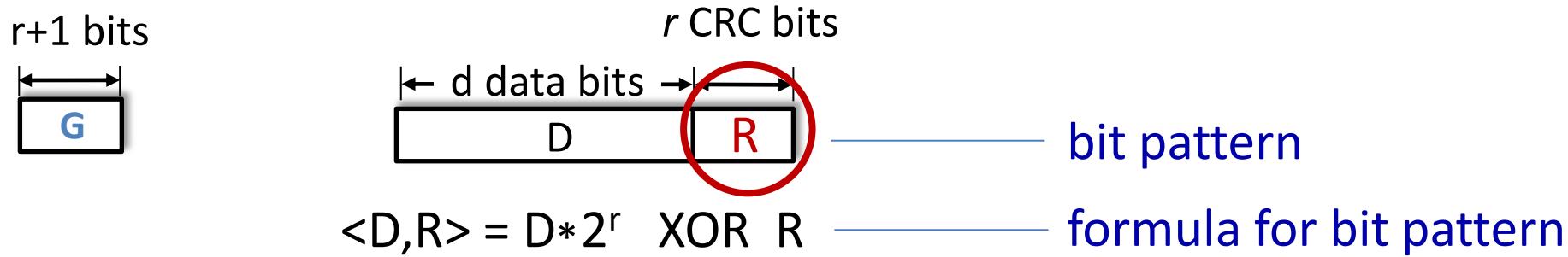


- Introduction to the link layer
- Error-Detection, Error-Correction
  - Parity Checks
  - Cyclic Redundancy Check (CRC)
- Multiple access protocols (MAC)
  - Channel Partitioning
  - Random Access Protocols
  - Taking-Turns Protocols
- Local Area Networks (LANs)
- Virtual Networks (VLANs)
- Data Centre Networking



# Cyclic Redundancy Check (CRC)

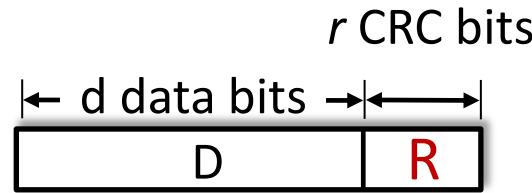
- More powerful error-detection coding
- **D**: data bits (given, think of these as a binary number)
- **G**: bit pattern (generator), of  $r+1$  bits (given)



**Goal:** choose  $r$  CRC bits, **R**, such that  $\langle D, R \rangle$  exactly divisible by G (mod 2)

- Receiver knows G, divides  $\langle D, R \rangle$  by G. If non-zero remainder: error detected!
- Can detect all burst errors less than  $r+1$  bits
- Widely used in practice (Ethernet, 802.11 WiFi)

# Cyclic Redundancy Check (CRC)



$$D \cdot 2^r \text{XOR } R = nG$$

$$D \cdot 2^r = nG \text{ XOR } R$$

$$\frac{D \cdot 2^r}{G} = n \text{ XOR } R$$

$$R = \text{remain.} \frac{D \cdot 2^r}{G}$$



- We choose  $R$  such that  $G$  divides into  $D \cdot 2^r$  XOR  $R$  without remainder.
- XOR  $R$  on both sides of the eq.
- Divide both sides by  $G$
- What is left in that division is our desired  $R$ .

$D=10011110$ ,  $r=3$ ,  
 $G=1001$  ( $r+1$  bit)

Calculate R  
10011110

Check for error  
10011110 111

00001110 000

00001110 111

00000111 000

00000111 111

00000011 100

00000011 011

00000001 110

00000001 001

00000000 111

00000000 000

# Example continued...

D=10011110, r=3,  
G=1001 (r+1 bit)

|                     | Check for error<br>(no error) | Check for error<br>(1 error) | Check for error<br>(2 errors)        | Check for error<br>(4 errors)         |
|---------------------|-------------------------------|------------------------------|--------------------------------------|---------------------------------------|
| Calculate R         |                               |                              |                                      |                                       |
| 10011110 000        | 10011110 111                  | 10 <u>1</u> 11110 <b>111</b> | 10 <u>1</u> 1111 <u>1</u> <b>111</b> | 10 <u>1</u> 1 <u>00</u> 11 <b>111</b> |
| 1001                | 1001                          |                              | 1001                                 | 1001                                  |
| 00001110 000        | 00001110 111                  | 00101110 <b>111</b>          | 00101111 <b>111</b>                  | 00100011 <b>111</b>                   |
| 1001                | 1001                          |                              | 1001                                 | 1001                                  |
| 00000111 000        | 00000111 111                  | 00001010 <b>111</b>          | 00001011 <b>111</b>                  | 00000111 <b>111</b>                   |
| 100 1               | 100 1                         |                              | 1001                                 | 100 1                                 |
| 00000011 100        | 00000011 011                  | 00000011 <b>111</b>          | 00000010 <b>111</b>                  | 00000011 <b>011</b>                   |
| 10 01               | 10 01                         |                              | 10 01                                | 10 01                                 |
| 00000001 110        | 00000001 001                  | 00000001 <b>101</b>          | 00000000 <b>101</b>                  | 00000001 <b>001</b>                   |
| 1 001               | 1 001                         |                              |                                      | 10 01                                 |
| 00000000 <b>111</b> | 00000000 <b>000</b>           | 00000000 <b>100</b>          |                                      | 00000000 <b>000</b>                   |

# Another Example

---

10011001 000

1001

00001001 000

1001

00000000 000

# Notation of CRC codes

---

$$G_{\text{CRC-32}} = 100000100110000010001110110110111$$

$$\text{CRC-32 : } x^{32} + x^{26} + x^{23} + x^{22} + x^{16} + x^{12} + x^{11} + x^{10} + x^8 + x^7 + x^5 + x^4 + x^2 + x + 1$$

$$\text{CRC-4 : } x^4 + x + 1$$

$$G_{\text{CRC-4}} = 10011$$

Examples of CRC representations

| Name  | Normal | Reversed | Reversed reciprocal |
|-------|--------|----------|---------------------|
| CRC-4 | 0x3    | 0xC      | 0x9                 |

- $0x3 = 0b0011$ , representing  $x^4 + (0x^3 + 0x^2 + 1x^1 + 1x^0)$  (MSB-first code)
- $0xC = 0b1100$ , representing  $(1x^0 + 1x^1 + 0x^2 + 0x^3) + x^4$  (LSB-first code)
- $0x9 = 0b1001$ , representing  $(1x^4 + 0x^3 + 0x^2 + 1x^1) + x^0$  (Koopman notation)

# Time for a break

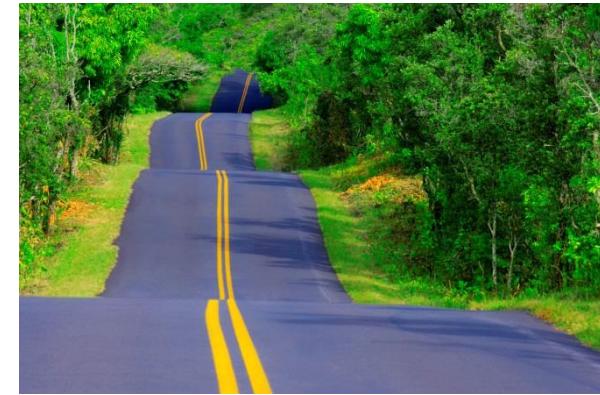
---



Take a break, maybe a coffee or tea, and continue when you are back.

# Roadmap Data-Link Layer

- Introduction to the link layer
- Error-Detection, Error-Correction
  - Parity Checks
  - Cyclic Redundancy Check (CRC)
- Multiple access protocols (MAC)
  - Channel Partitioning
  - Random Access Protocols
  - Taking-Turns Protocols
- Local Area Networks (LANs)
- Virtual Networks (VLANs)
- Data Centre Networking



# Multiple access links, protocols

Two types of “links”:

- Point-to-point
  - point-to-point link between Ethernet switch, host
- Broadcast (shared wire or medium)
  - old-fashioned Ethernet
  - upstream HFC in cable-based access network
  - 802.11 wireless LAN, 4G/5G. satellite



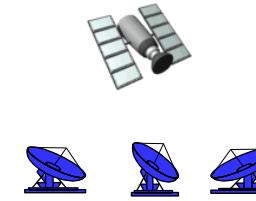
shared wire (e.g.,  
Internet via TV cable)



shared radio: 4G/5G



shared radio: WiFi



shared radio: satellite



humans at a cocktail party  
(shared air, acoustical)

# Multiple access protocols

---

- Single shared broadcast channel
- Two or more simultaneous transmissions by nodes: interference
  - *collision* if node receives two or more signals at the same time

## multiple access protocol

- distributed algorithm that determines how nodes share channel, i.e., determine when node can transmit
- communication about channel sharing must use channel itself!
  - no out-of-band channel for coordination

# An ideal multiple access protocol

---

*given:* Multiple access channel (MAC) of rate  $R$  bps

*Desiderata (what we want):*

1. When one node wants to transmit, it can send at rate  $R$ .
2. When  $M$  nodes want to transmit, each can send at average rate  $R/M$
3. Fully decentralized:
  - no special node to coordinate transmissions
  - no synchronization of clocks, slots
4. Simple

# MAC protocols: Taxonomy

---

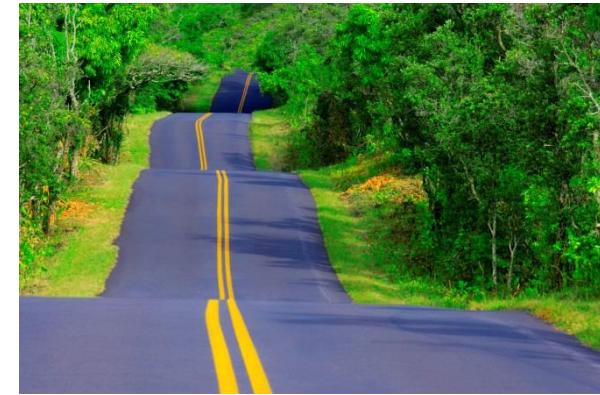
Three broad classes:

- **Channel partitioning**
  - Divide channel into smaller “pieces” (time slots, frequency, code)
  - Allocate piece to node for exclusive use
- ***Random access***
  - Channel not divided, allow collisions
  - “Recover” from collisions
- **“Taking turns”**
  - Nodes take turns, but nodes with more to send can take longer turns

# Roadmap Data-Link Layer



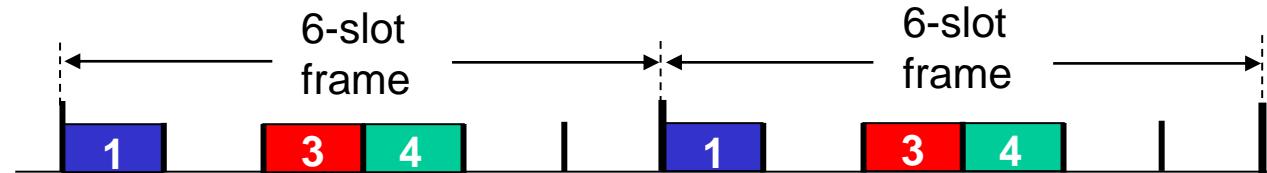
- Introduction to the link layer
- Error-Detection, Error-Correction
  - Parity Checks
  - Cyclic Redundancy Check (CRC)
- **Multiple access protocols (MAC)**
  - Channel Partitioning
  - Random Access Protocols
  - Taking-Turns Protocols
- Local Area Networks (LANs)
- Virtual Networks (VLANs)
- Data Centre Networking



# Channel partitioning MAC protocols: TDMA

## TDMA: Time Division Multiple Access

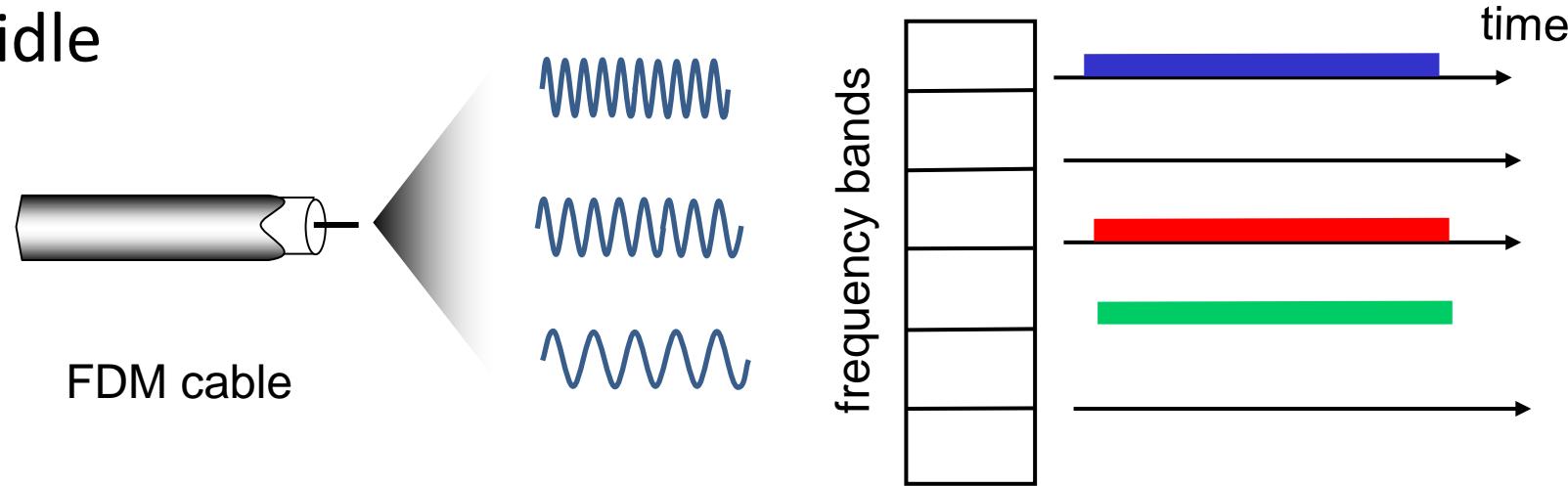
- Access to channel in “rounds”
- Each station gets fixed length slot (length = packet transmission time) in each round
- Unused slots go idle
- Example: 6-station LAN, 1,3,4 have packets to send, slots 2,5,6 idle



# Channel partitioning MAC protocols: FDMA

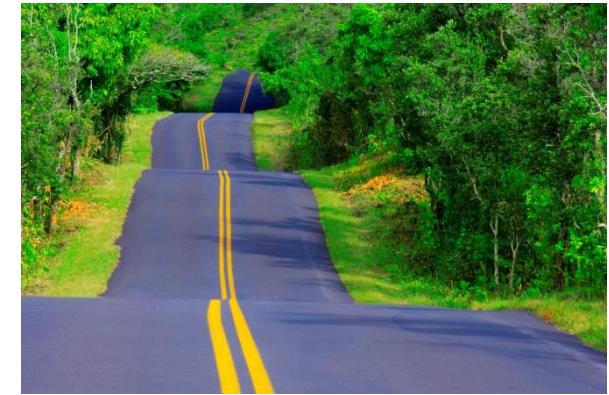
## FDMA: Frequency Division Multiple Access

- Channel spectrum divided into frequency bands
- Each station assigned fixed frequency band
- Unused transmission time in frequency bands go idle
- Example: 6-station LAN, 1,3,4 have packet to send, frequency bands 2,5,6 idle



# Roadmap Data-Link Layer

- Introduction to the link layer
- Error-Detection, Error-Correction
  - Parity Checks
  - Cyclic Redundancy Check (CRC)
- **Multiple access protocols (MAC)**
  - Channel Partitioning
  - Random Access Protocols
  - Taking-Turns Protocols
- Local Area Networks (LANs)
- Virtual Networks (VLANs)
- Data Centre Networking



# Random access protocols

---

- When node has packet to send
  - transmit at full channel data rate  $R$ .
  - no *a priori* coordination among nodes.
- Two or more transmitting nodes: “**collision**”
- Random access MAC protocol specifies:
  - How to detect collisions
  - How to recover from collisions (e.g., via delayed retransmissions)
- Examples of random access MAC protocols:
  - ALOHA, slotted ALOHA
  - CSMA (Carrier Sense Multiple Access): CSMA/CD, CSMA/CA

# Slotted ALOHA

---

## Assumptions:

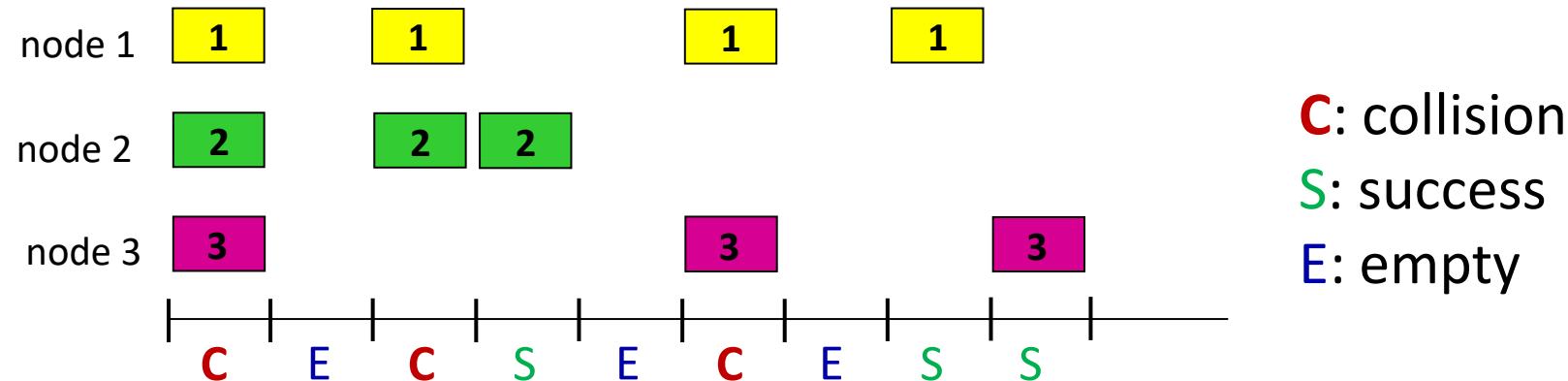
- All frames same size
- Time divided into equal size slots (time to transmit 1 frame)
- Nodes start to transmit only slot beginning
- Nodes are synchronized
- If 2 or more nodes transmit in slot, all nodes detect collision

## Operation:

- When node obtains fresh frame, transmits in next slot
  - *If no collision*: node can send new frame in next slot
  - *If collision*: node retransmits frame in each subsequent slot with probability  $p$  until success

randomization – why?

# Slotted ALOHA



## Pros:

- Single active node can continuously transmit at full rate of channel
- Highly decentralized: only slots in nodes need to be in sync
- Simple 😊

## Cons:

- Collisions, wasting slots
- Idle slots
- Nodes may be able to detect collision in less than time to transmit packet
- Clock synchronization

# Slotted ALOHA: Efficiency

---

**Efficiency:** long-run fraction of successful slots (many nodes, all with many frames to send)

- *Suppose:*  $N$  nodes with many frames to send, each transmits in slot with probability  $p$

- prob that given node has success in a slot =  $p(1-p)^{N-1}$
- prob that *any (of  $N$ )* node has a success =  $Np(1-p)^{N-1}$
- max efficiency: find  $p^*$  that maximizes  $Np(1-p)^{N-1}$
- for many nodes, take limit of  $Np^*(1-p^*)^{N-1}$  as  $N$  goes to infinity, gives:

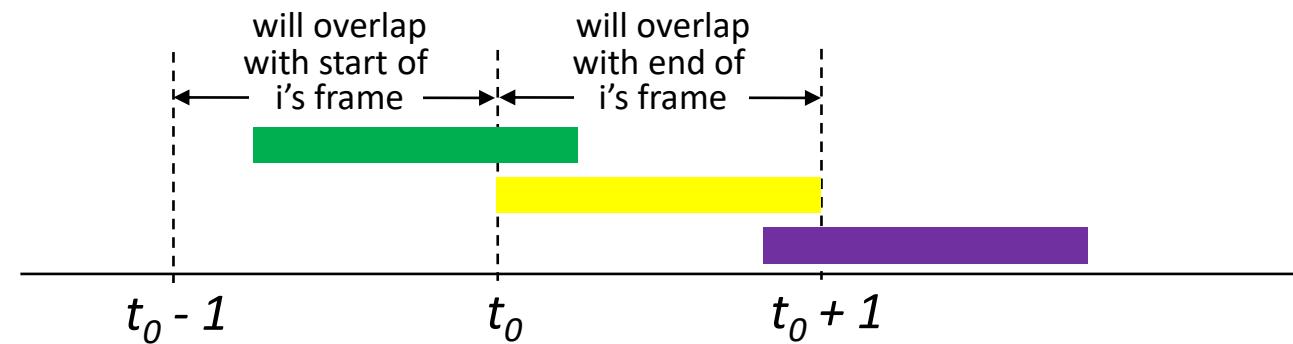
$$\lim_{N \rightarrow \infty} Np^*(1 - p^*)^{N-1} = \frac{1}{e} = 0.37$$

- *At best:* channel used for useful transmissions 37% of time!

# Pure ALOHA

---

- Unslotted Aloha: simpler, no synchronization
- Collision probability increases with no synchronization:
  - Frame sent at  $t_0$  collides with other frames sent in  $[t_0 - 1, t_0 + 1]$



# Pure ALOHA: Efficiency

---

$$\begin{aligned} P(\text{success by given node}) &= P(\text{node transmits}) * \\ &\quad P(\text{no other node transmits in } [t_0-1, t_0] *) * \\ &\quad P(\text{no other node transmits in } [t_0, t_0+1]) \\ &= p \cdot (1-p)^{N-1} \cdot (1-p)^{N-1} \\ &= p \cdot (1-p)^{2(N-1)} \end{aligned}$$

... choosing optimum  $p^*$  and then letting  $N \rightarrow \infty$

$$\lim_{N \rightarrow \infty} p^* \cdot (1 - p^*)^{2(N-1)} = \frac{1}{2e} = 0.184$$

even worse than slotted Aloha!

# CSMA (carrier sense multiple access)

---

Simple **CSMA**: listen before transmit:

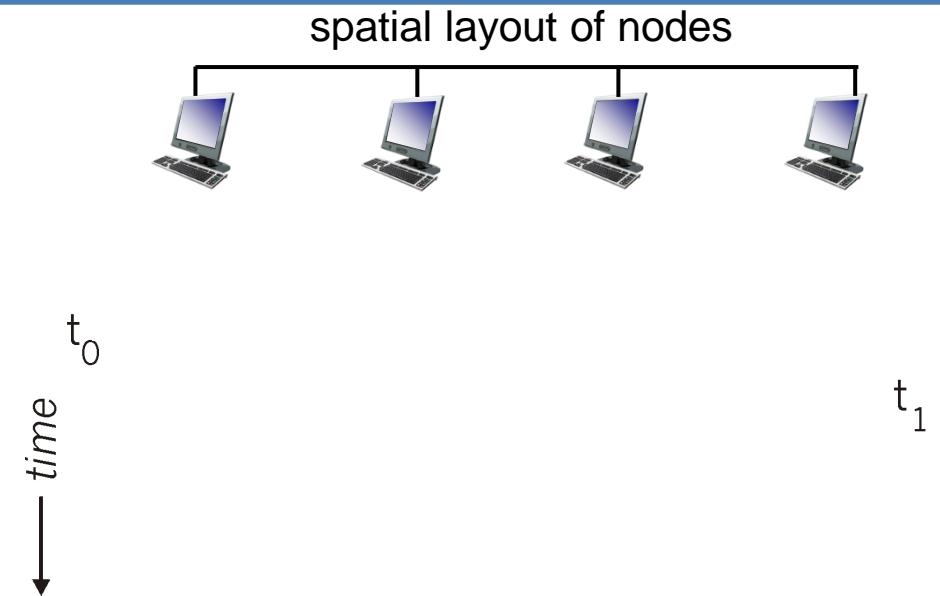
- if channel sensed idle: transmit entire frame
- if channel sensed busy: defer transmission
- Human analogy: **don't interrupt others!**

**CSMA/CD**: CSMA with *collision detection*

- Collisions *detected* within short time
- Colliding transmissions aborted, reducing channel wastage
- Collision detection easy in wired, difficult with wireless
- Human analogy: the polite conversationalist

# CSMA: collisions

- Collisions *can* still occur with carrier sensing:
  - Propagation delay means two nodes may not hear each other's just-started transmission
- Collision: entire packet transmission time wasted
  - Distance & propagation delay play role in determining collision probability
- CSMA/CD reduces the amount of time wasted in collisions
  - transmission aborted on collision detection



# Ethernet CSMA/CD algorithm

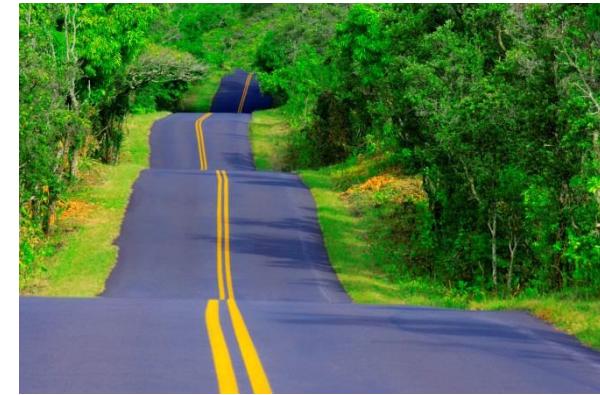
---

1. NIC receives datagram from network layer, creates frame
2. If NIC senses channel:
  - if **idle**: start frame transmission.
  - if **busy**: wait until channel idle, then transmit
3. If NIC transmits entire frame without collision, NIC is done with frame !
4. If NIC detects another transmission while sending: abort, send jam signal
5. After aborting, NIC enters ***binary (exponential) backoff***:
  - after  $m$ th collision, NIC chooses  $K$  at random from  $\{0,1,2, \dots, 2^m-1\}$ . NIC waits  $K \cdot 512$  bit times, returns to Step 2
  - more collisions: longer backoff interval

# Roadmap Data-Link Layer



- Introduction to the link layer
- Error-Detection, Error-Correction
  - Parity Checks
  - Cyclic Redundancy Check (CRC)
- **Multiple access protocols (MAC)**
  - Channel Partitioning
  - Random Access Protocols
  - Taking-Turns Protocols
- Local Area Networks (LANs)
- Virtual Networks (VLANs)
- Data Centre Networking



# “Taking turns” MAC protocols

---

## Channel partitioning MAC protocols:

- Share channel *efficiently* and *fairly* at high load
- Inefficient at low load: delay in channel access,  $1/N$  bandwidth allocated even if only 1 active node!

## Random access MAC protocols:

- Efficient at low load: single node can fully utilize channel
- High load: collision overhead

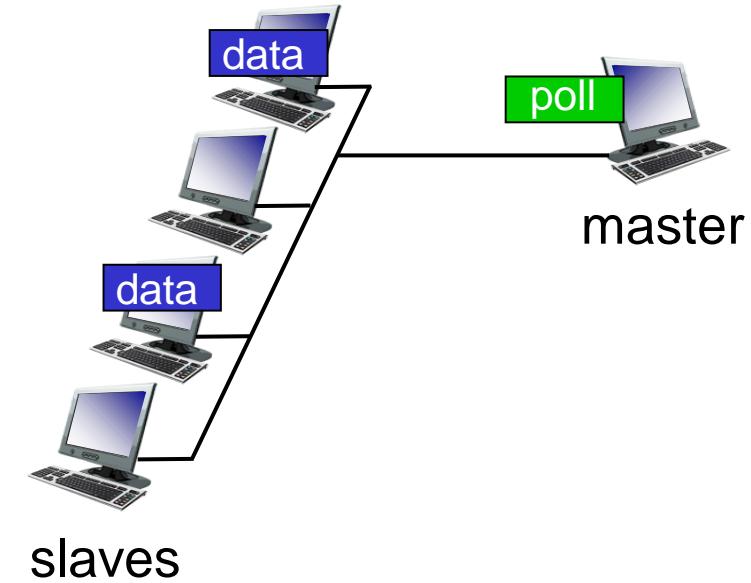
## “Taking turns” protocols:

- Look for best of both worlds!
- For example: Bluetooth protocol

# “Taking turns” MAC protocols

## Polling:

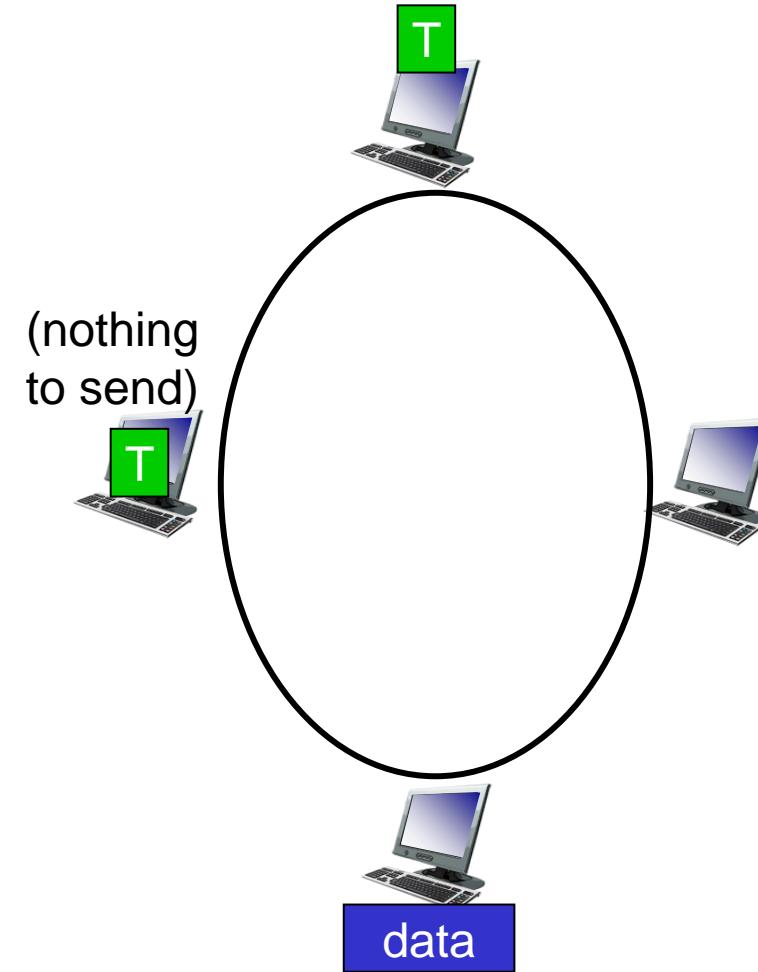
- Master node “invites” other nodes to transmit in turn
- Typically used with “dumb” devices (Bluetooth)
- Concerns:
  - polling overhead
  - latency
  - single point of failure (master)



# “Taking turns” MAC protocols

## Token passing:

- Control *token* passed from one node to next sequentially.
- Token message
- Concerns:
  - token overhead
  - latency
  - single point of failure (token)



# Summary of MAC protocols

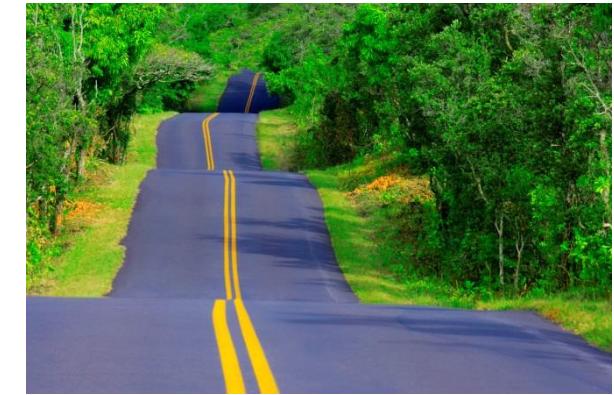
---

- **Channel partitioning**, by time, frequency or code
  - Time Division, Frequency Division
- **Random access (dynamic)**,
  - ALOHA, S-ALOHA, CSMA, CSMA/CD
  - carrier sensing: easy in some technologies (wire), hard in others (wireless)
  - CSMA/CD used in Ethernet
  - CSMA/CA used in 802.11
- **Taking turns**
  - polling from central site, token passing
  - Bluetooth, FDDI, token ring

# Thanks!

---

- Introduction to the link layer
- Error-Detection, Error-Correction
  - Parity Checks
  - Cyclic Redundancy Check (CRC)
- Multiple access protocols (MAC)
  - Channel Partitioning
  - Random Access Protocols
  - Taking-Turns Protocols
- Local Area Networks (LANs)
- Virtual Networks (VLANs)
- Data Centre Networking





# Course on Computer Communication and Networks

## Lecture 9 Chapter 6; Link Layer Part 2

EDA344/DIT423/LEU062

Lecturer: Hans-Martin Heyn

Based on the book Computer Networking: A Top Down Approach, Jim Kurose, Keith Ross, Addison-Wesley.

# Layering: The OSI Reference Model

ISO (International Standards Organization) defined the OSI (Open Systems Interconnect) model to help vendors create interoperable network implementation

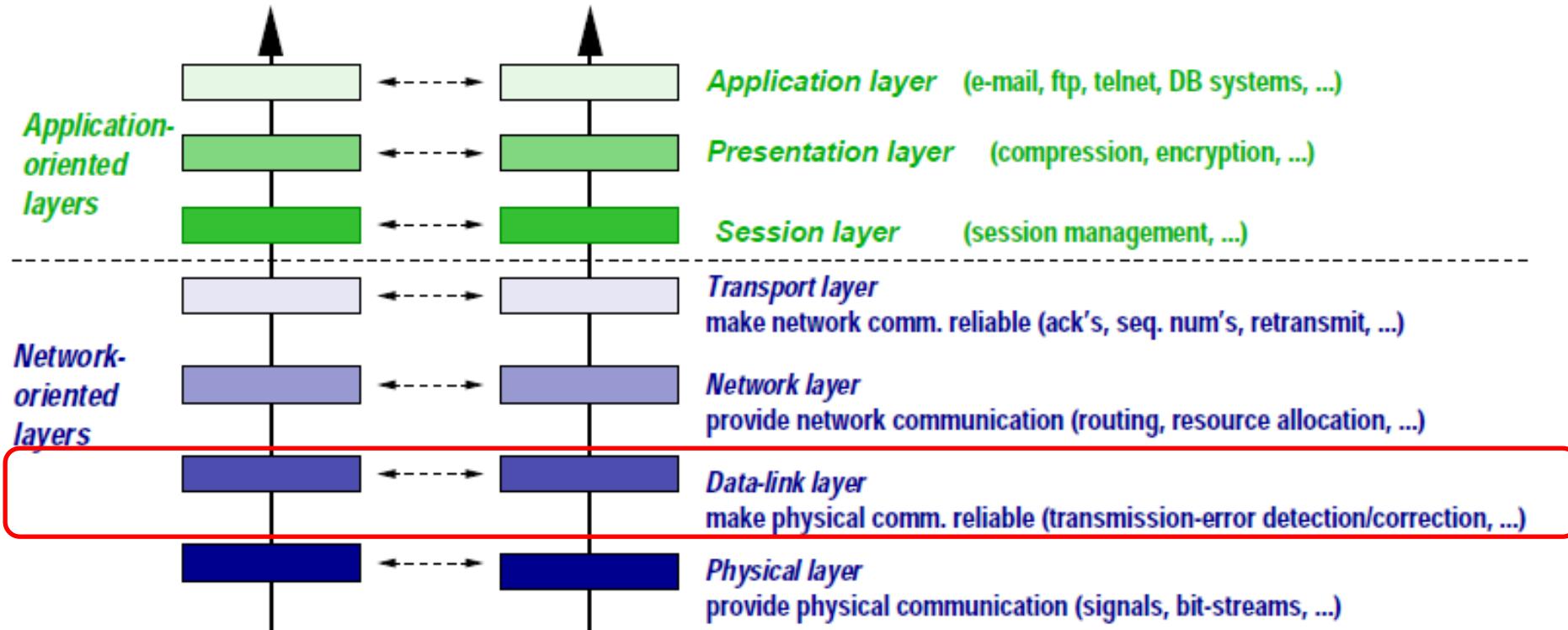
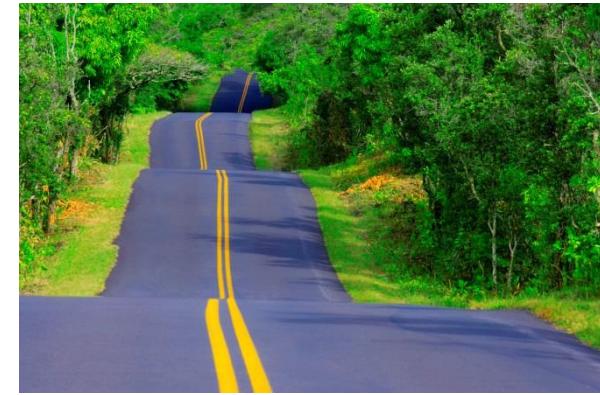


Fig. Steen, Sips : Computer and Network organization

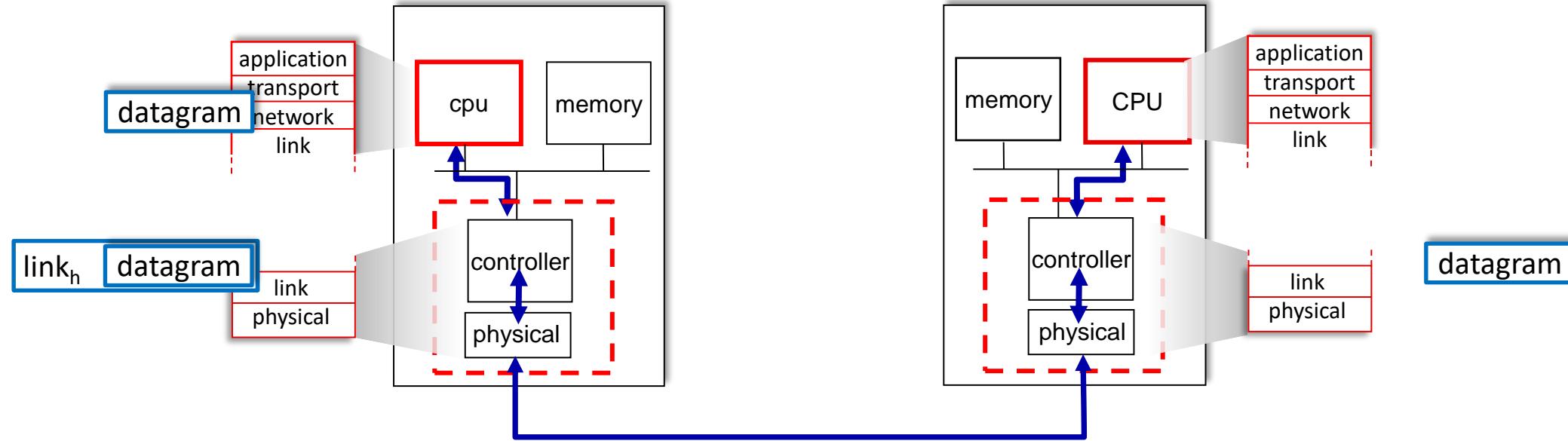
``X dot'' series (X.25, X. 400, X.500) OSI model implementation (protocol stack)

# Roadmap Data-Link Layer

- Introduction to the link layer
- Error-Detection, Error-Correction
- Multiple access protocols (MAC)
- Local Area Networks (LANs)
  - Addressing, Address resolution protocol (ARP)
  - Ethernet
  - Switches
- Virtual Networks (VLANs)
- Data Centre Networking



# Recap: Interfaces communicating



**Sending side:**

- encapsulates datagram in frame
- adds error checking bits, reliable data transfer, flow control, etc.

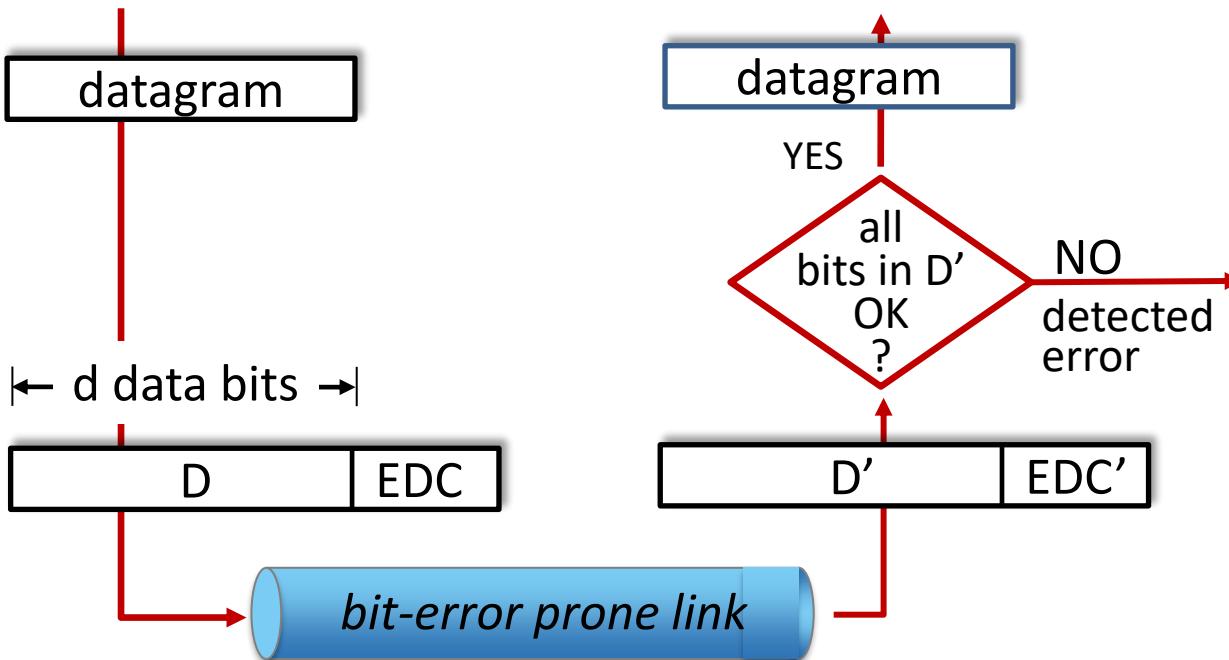
**Receiving side:**

- looks for errors, reliable data transfer, flow control, etc.
- extracts datagram, passes to upper layer at receiving side

# Recap: Error detection

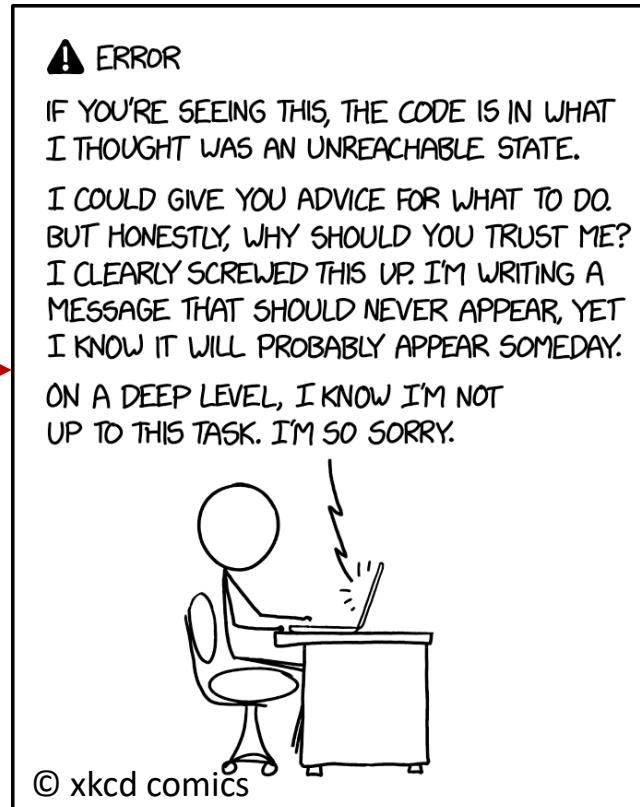
**EDC:** Error detection and correction bits (e.g., redundancy)

**D:** Data protected by error checking, may include header fields



Error detection not 100% reliable!

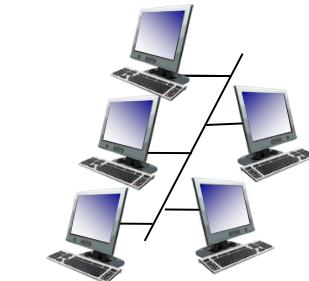
- protocol may miss some errors, but rarely
- larger EDC field yields better detection and correction



# Recap: Multiple access links, protocols

Two types of “links”:

- Point-to-point
  - point-to-point link between Ethernet switch, host
  - PPP for dial-up access
- Broadcast (shared wire or medium)
  - old-fashioned Ethernet
  - upstream HFC in cable-based access network
  - 802.11 wireless LAN, 4G/5G. satellite



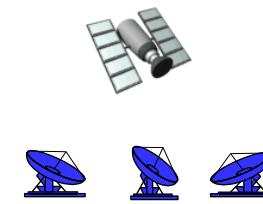
shared wire (e.g.,  
Internet via TV cable)



shared radio: 4G/5G



shared radio: WiFi



shared radio: satellite



humans at a cocktail party  
(shared air, acoustical)

# Recap: MAC (Medium Access Control) protocols

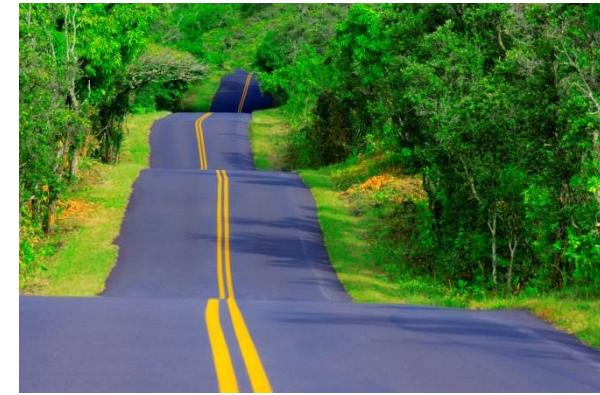
---

Three broad classes:

- **Channel partitioning**
  - Divide channel into smaller “pieces” (time slots, frequency, code)
  - Allocate piece to node for exclusive use
- **Random access**
  - Channel not divided, allow collisions
  - “Recover” from collisions
- **“Taking turns”**
  - Nodes take turns, but nodes with more to send can take longer turns

# Roadmap Data-Link Layer

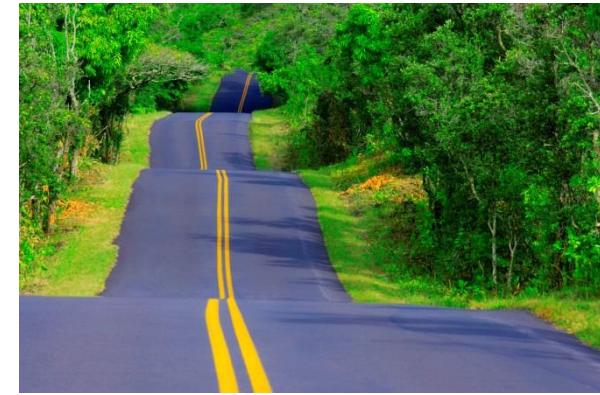
- Introduction to the link layer
- Error-Detection, Error-Correction
- Multiple access protocols (MAC)
- Local Area Networks (LANs)
  - Addressing, Address resolution protocol (ARP)
  - Ethernet
  - Switches
- Virtual Networks (VLANs)
- Data Centre Networking



# Roadmap Data-Link Layer



- Introduction to the link layer
- Error-Detection, Error-Correction
- Multiple access protocols (MAC)
- Local Area Networks (LANs)
  - Addressing, Address resolution protocol (ARP)
  - Ethernet
  - Switches
- Virtual Networks (VLANs)
- Data Centre Networking



# Media Access Control (MAC) addresses

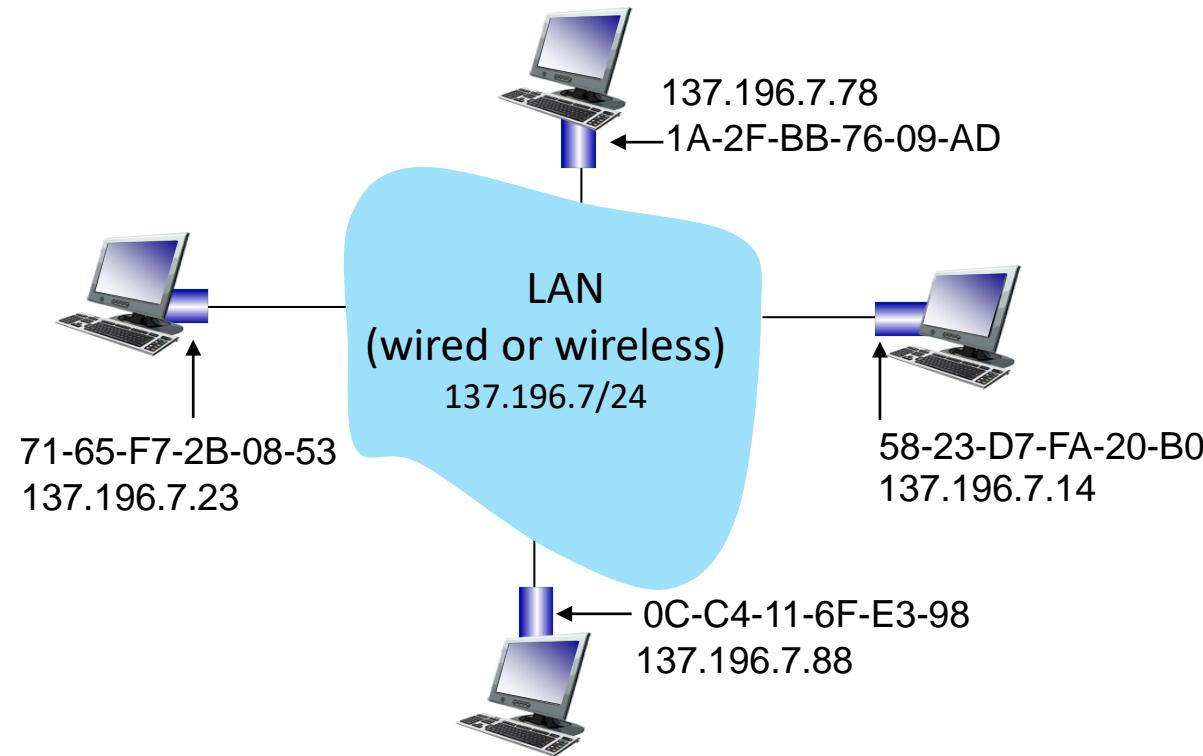
---

- 32-bit / 128-bit IP address:
    - *network-layer* address for interface
    - used for layer 3 (network layer) forwarding and routing
    - e.g.: 128.119.40.136, or 2a02:ab1:100c:f9b8:e18e:2327:a326:18b3
  - MAC (or LAN or physical or Ethernet) address:
    - function: used “locally” to get frame from one interface to another physically-connected interface (same subnet, in IP-addressing sense)
    - 48-bit MAC address (for most LANs) burned in NIC ROM, also sometimes software settable
    - e.g.: 1A-2F-BB-76-09-AD
- hexadecimal (base 16) notation  
(each “numeral” represents 4 bits)*

# Media Access Control (MAC) addresses

each interface on LAN

- has unique, typically permanent 48-bit **MAC** address
- has a locally unique but not necessarily permanent 32/128-bit IP address (as we've seen)



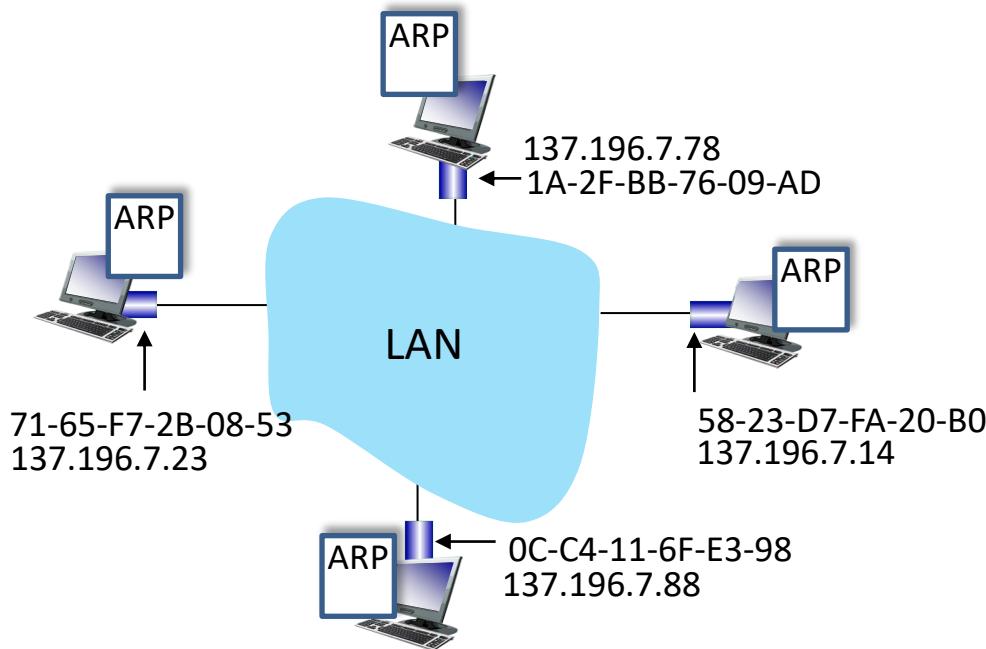
# Media Access Control (MAC) addresses

---

- MAC address allocation administered by IEEE (Institute of Electrical and Electronics Engineers in U.S.)
- Manufacturer buys portion of MAC address space (to assure uniqueness)
- Analogy:
  - MAC address: like Social Security Number
  - IP address: like postal address
- MAC flat address: portability
  - can move interface from one LAN to another
  - recall **IP address *not* portable**: depends on IP subnet to which node is attached

# ARP: Address Resolution Protocol

*Question:* Given an IP-address, how to determine corresponding interface's MAC address?



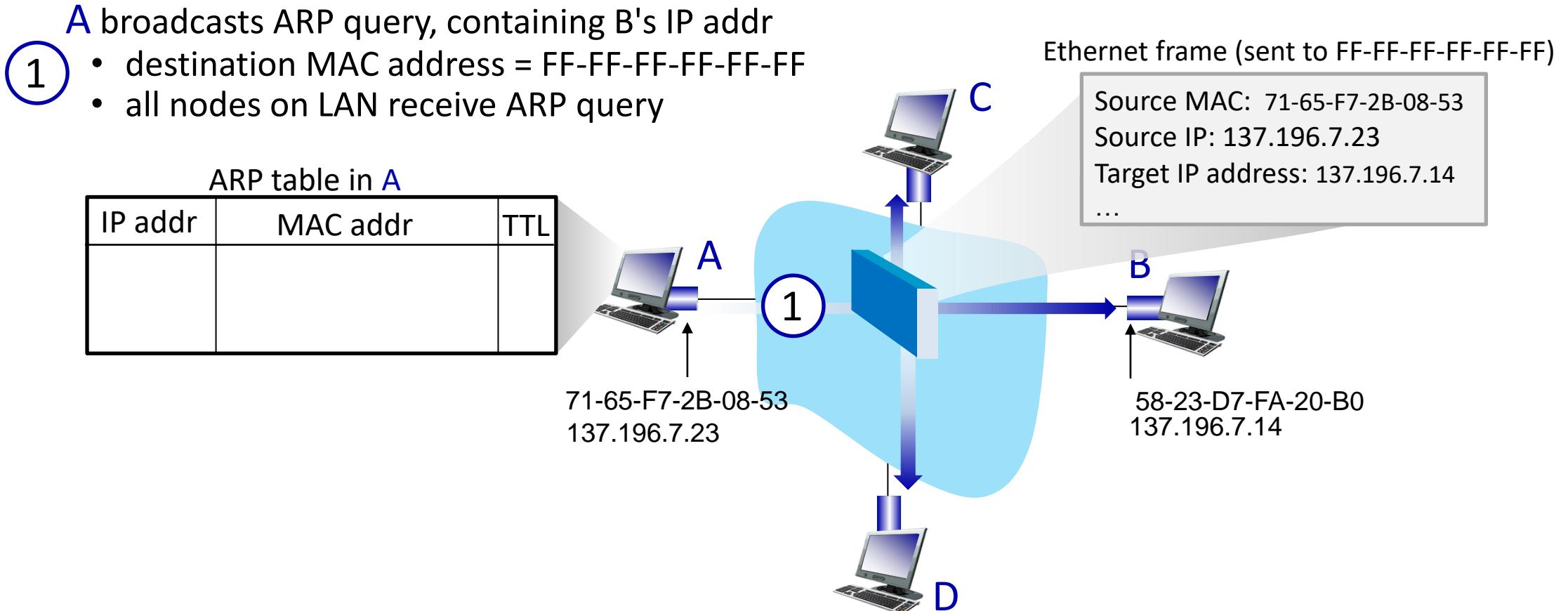
**ARP table:** each IP node (host, router) on LAN has an ARP table

- IP/MAC address mappings for some LAN nodes:  
<IP address; MAC address; TTL>
- TTL (Time To Live): time after which address mapping will be forgotten (typically 20 min)

# ARP protocol in action

Example: A wants to send datagram to B

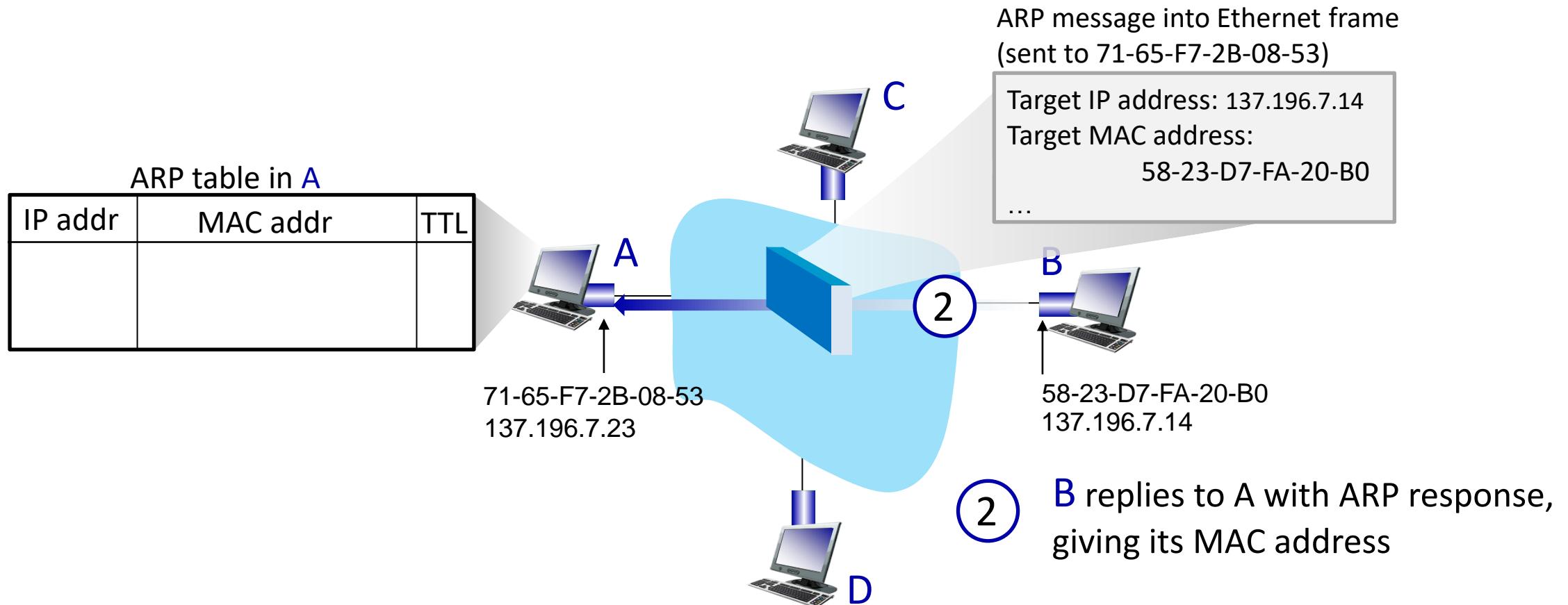
- B's MAC address not in A's ARP table, so A uses ARP to find B's MAC address



# ARP protocol in action

Example: A wants to send datagram to B

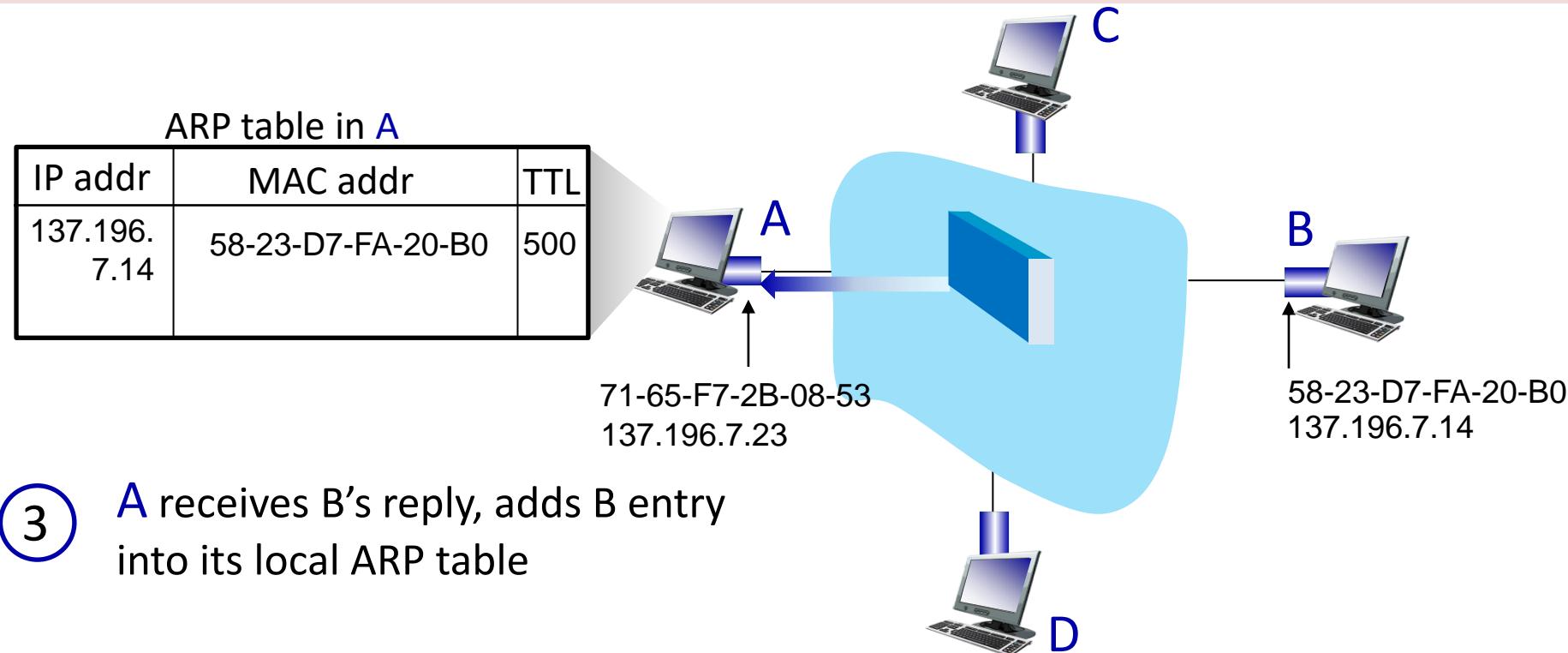
- B's MAC address not in A's ARP table, so A uses ARP to find B's MAC address



# ARP protocol in action

Example: A wants to send datagram to B

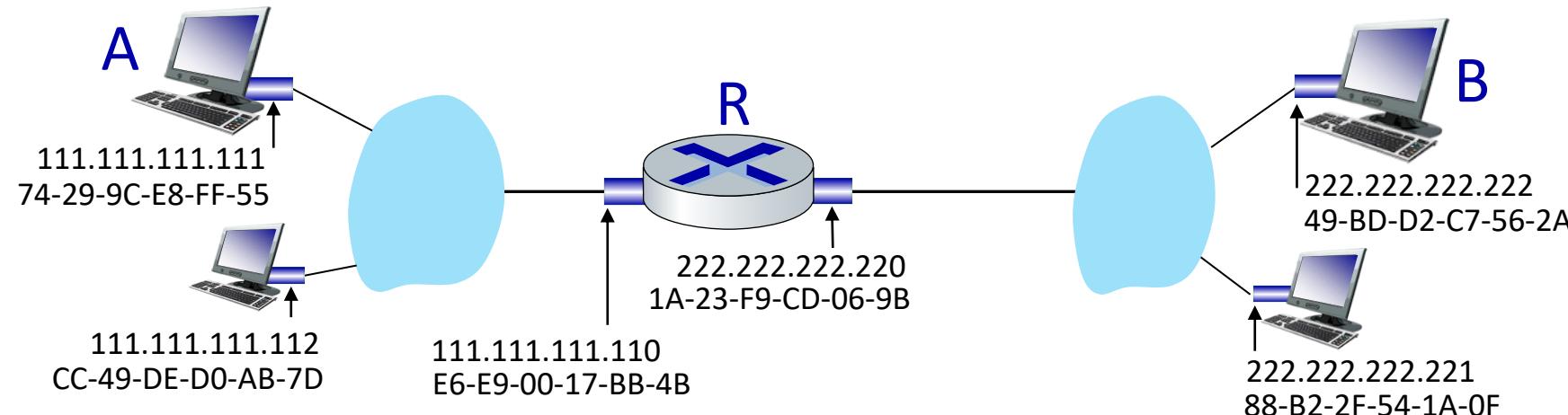
- B's MAC address not in A's ARP table, so A uses ARP to find B's MAC address
- A caches (saves) IP-to-MAC address pair in its ARP table until information becomes old (TTL)
- ARP is "plug-and-play"
  - Nodes create their ARP tables without intervention from administrator
- ARP table only contains IP-to-MAC address pairs **for the same LAN!**



# Routing to another subnet: Addressing

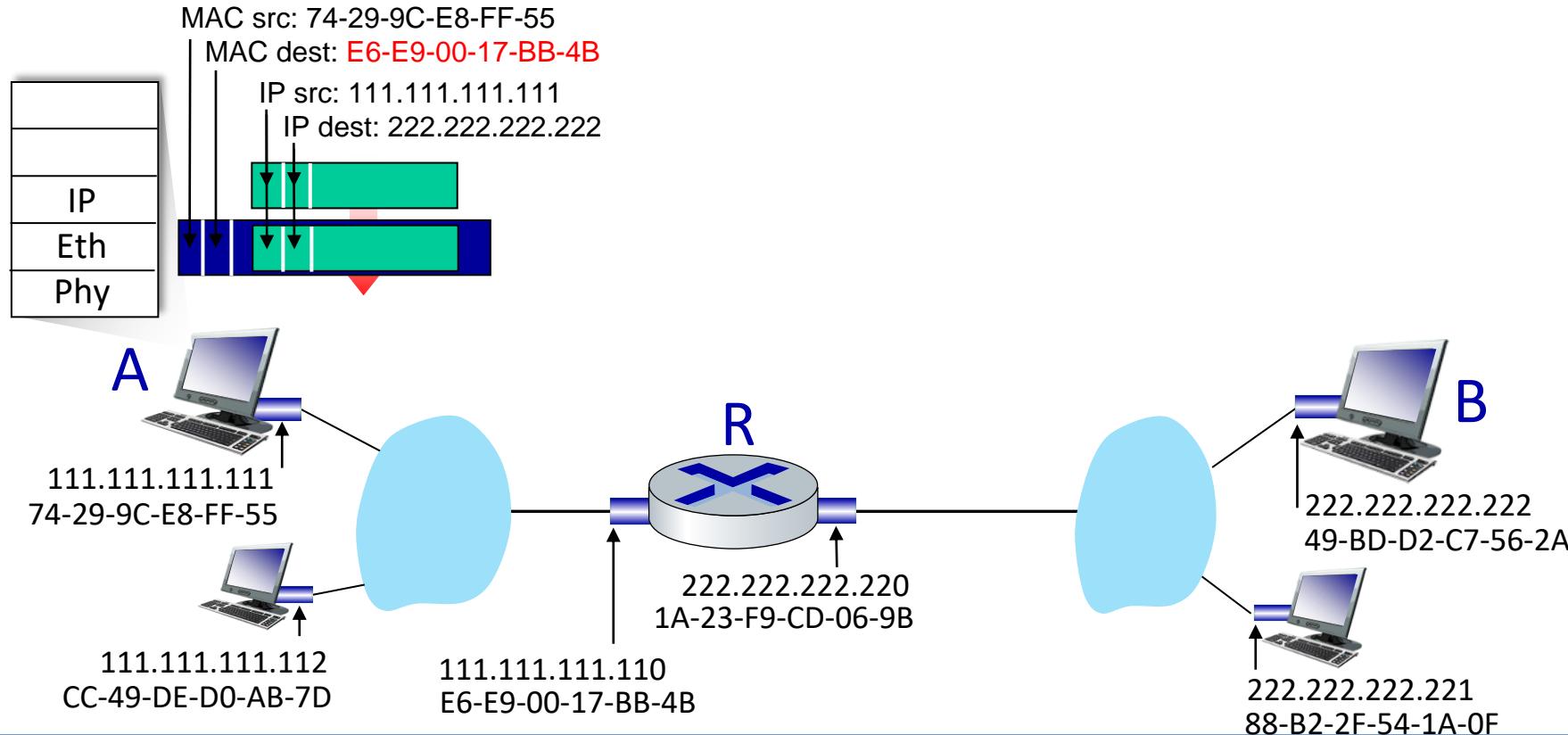
Walkthrough: sending a datagram from A to B via R

- Focus on addressing – at IP (datagram) and MAC layer (frame) levels
- Assume that:
  - A knows B's IP address
  - A knows IP address of first hop router, R
  - A knows R's MAC address



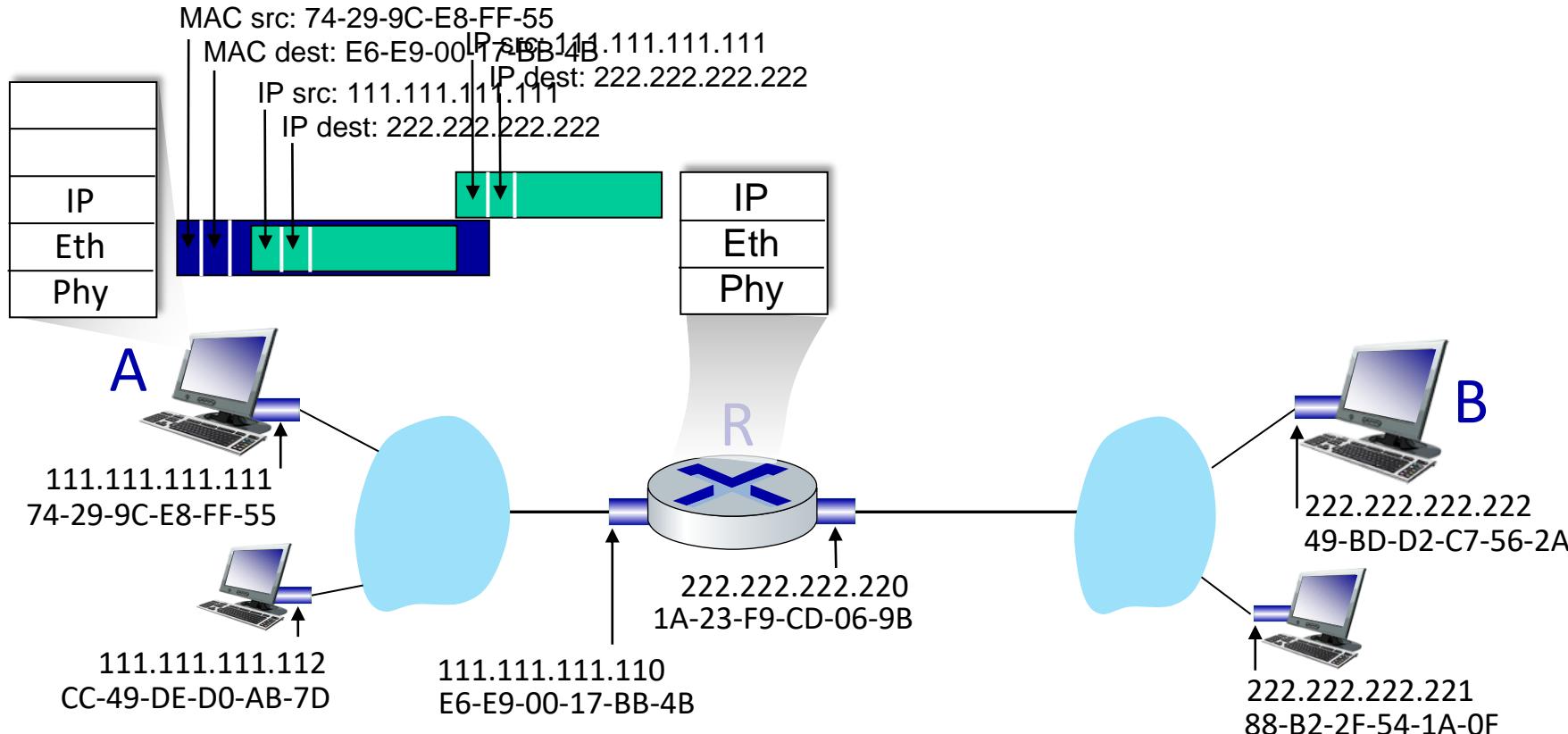
# Routing to another subnet: Addressing

- A creates IP datagram with IP address of source A and IP address of destination B
- A creates link-layer frame containing A-to-B IP datagram
  - **R's MAC address is frame's destination**



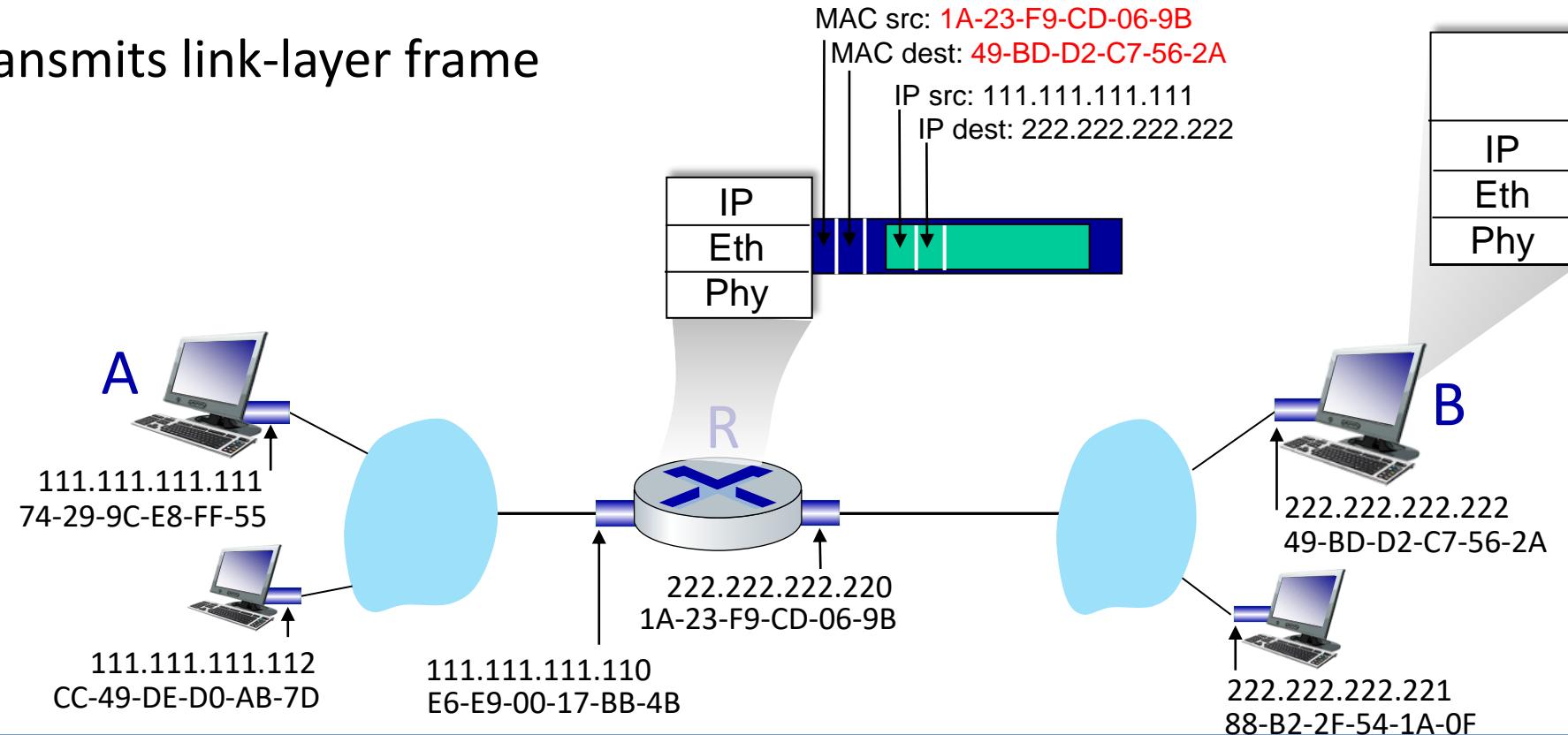
# Routing to another subnet: Addressing

- Frame sent from A to R
- Frame received at R, datagram removed, passed up to IP



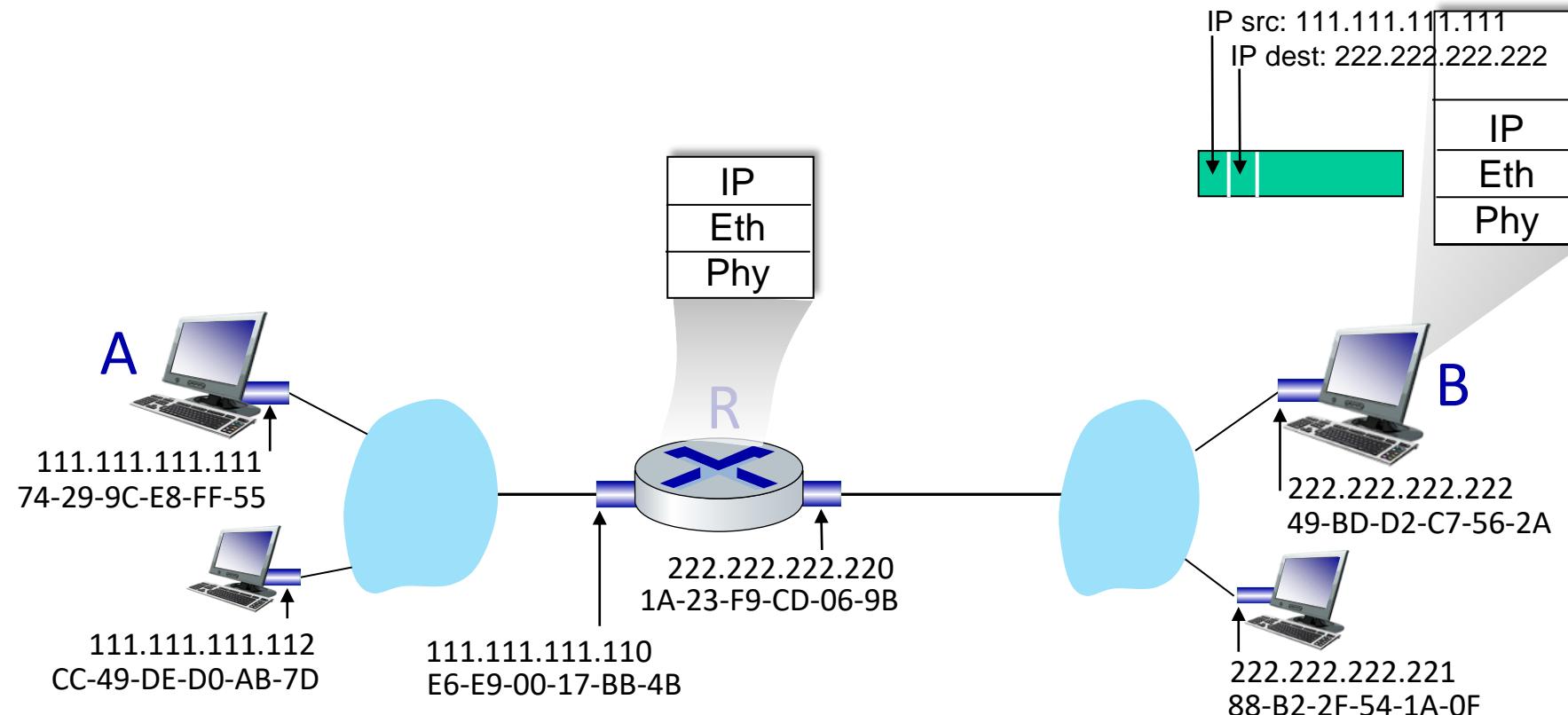
# Routing to another subnet: Addressing

- R determines outgoing interface, passes datagram with IP source A, destination B to link layer
- R creates link-layer frame containing A-to-B IP datagram. Frame destination address: B's MAC address
- Transmits link-layer frame



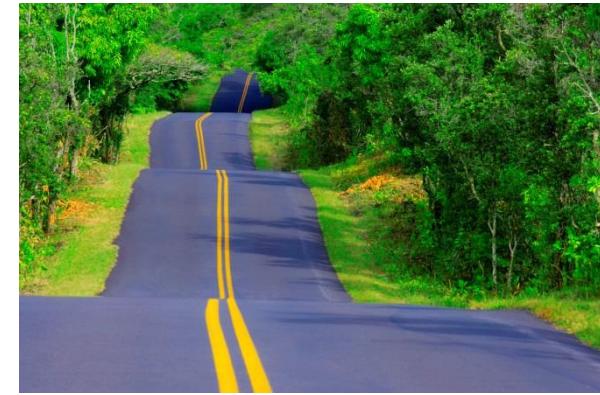
# Routing to another subnet: Addressing

- B receives frame, extracts IP datagram destination B
- B passes datagram up protocol stack to IP



# Roadmap Data-Link Layer

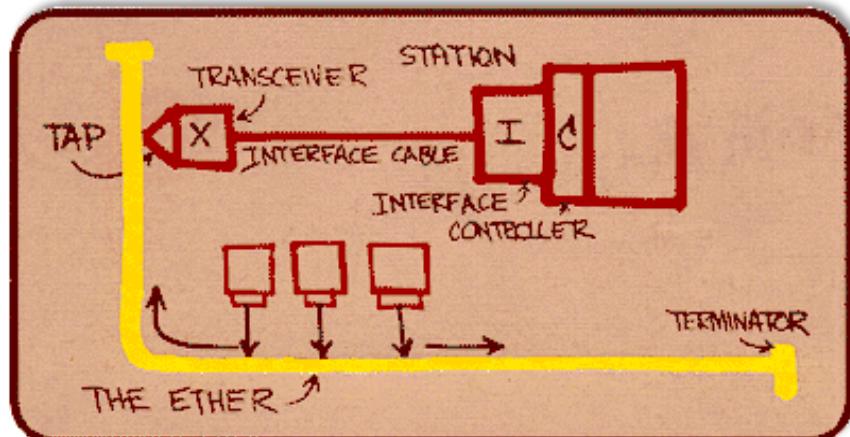
- Introduction to the link layer
- Error-Detection, Error-Correction
- Multiple access protocols (MAC)
- **Local Area Networks (LANs)**
  - Addressing, Address resolution protocol (ARP)
  - Ethernet
  - Switches
- Virtual Networks (VLANs)
- Data Centre Networking



# Ethernet

“Dominant” wired LAN technology:

- First widely used LAN technology
- Simple, cheap
- Kept up with speed race: 10 Mbps – 400 Gbps
- Single chip, multiple speeds (e.g., Broadcom BCM5761)



Metcalfe's Ethernet  
sketch

<https://www.uspto.gov/learning-and-resources/journeys-innovation/audio-stories/defying-doubters>

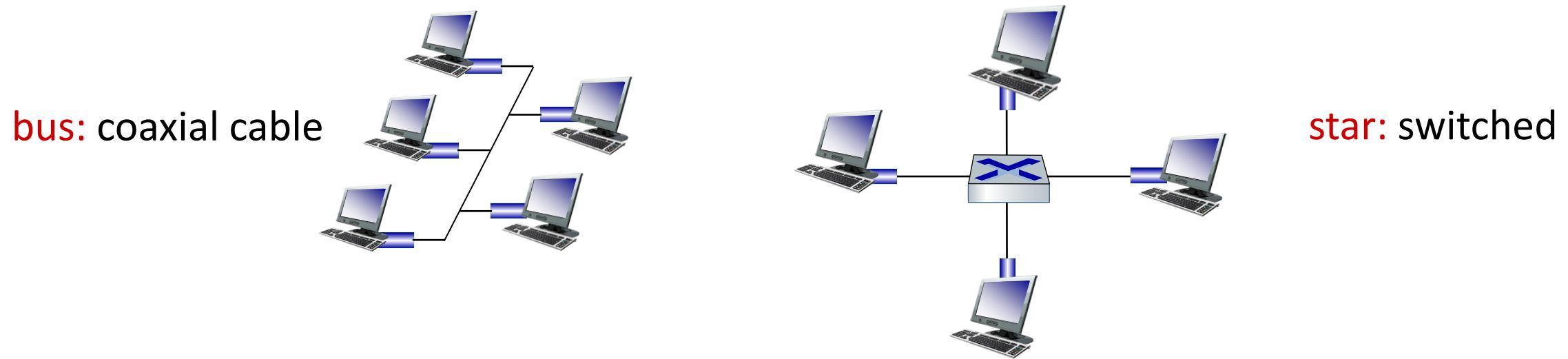
JACOBI BERNOULLI  
DISSERTATIO  
DE  
GRAVITATE  
ÆTHERIS.



AMSTELÆDAMI  
Apud HENR. WETSTENIUM  
clio Ioc LXXXIII.

# Ethernet: Physical topology

- **Bus:** Popular through mid 90s
  - All nodes in same collision domain (can collide with each other)
- **Switched:** Prevails today
  - Active link-layer *switch* in center
  - Nodes do not collide with each other



# Ethernet frame structure

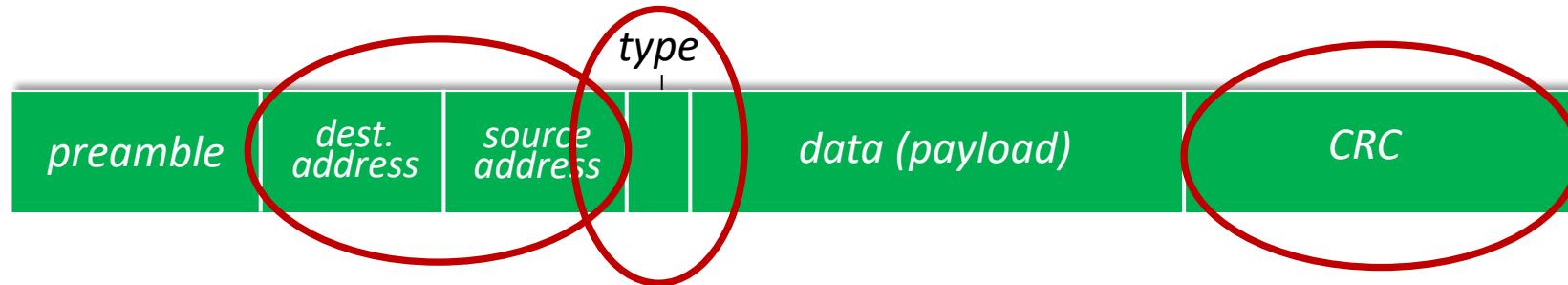
Sending interface **encapsulates** IP datagram (or other network layer protocol packet) **in Ethernet frame**



## *Preamble:*

- Used to synchronize receiver, sender clock rates
- 7 bytes of 10101010 followed by 1 byte of 10101011

# Ethernet frame structure

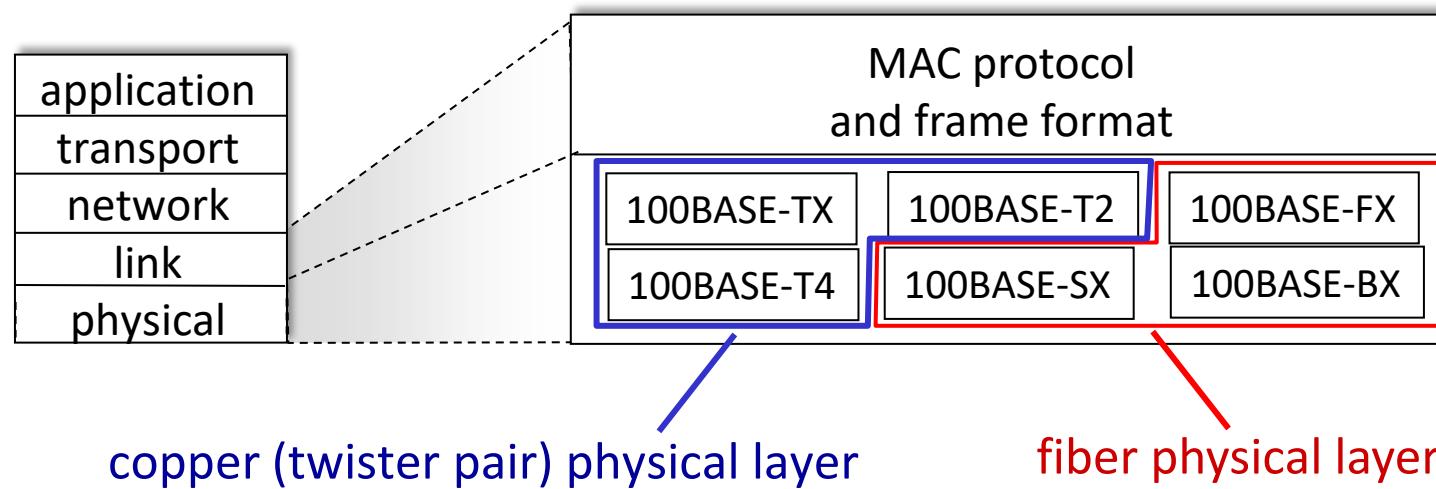


- **Addresses:** 6 byte source, destination MAC addresses
  - If adapter receives frame with matching destination address, or with broadcast address (e.g., ARP packet), it passes data in frame to network layer protocol
  - Otherwise, adapter discards frame
- **Type:** indicates higher layer protocol
  - Mostly IP but others possible, e.g., Novell IPX, AppleTalk
  - Used to demultiplex up at receiver
- **CRC:** cyclic redundancy check at receiver
  - Error detected: frame is dropped

# 802.3 Ethernet standards: link & physical layers

- *Many* different Ethernet standards

- Common MAC protocol and frame format
- Different speeds: 2 Mbps, 10 Mbps, 100 Mbps, 1Gbps, 10 Gbps, 40 Gbps
- Different physical layer media: Fiber, Cable (2 wires, 4 wires, 8 wires, shielding against electromagnetic interference)



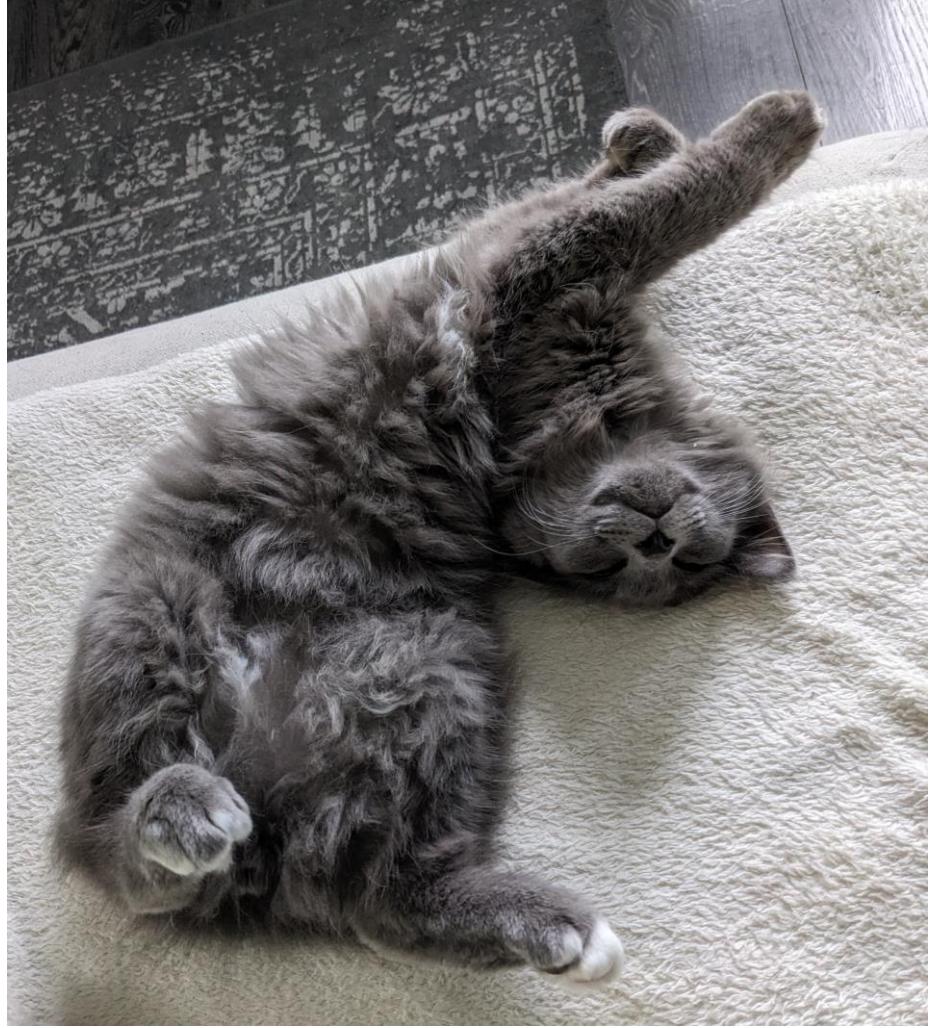
# Ethernet: Connectionless, Unreliable

---

- **Connectionless:** No handshaking between sending and receiving NICs
- **Unreliable:** Receiving NIC does not send ACKs or NAKs to sending NIC
  - data in dropped frames recovered only if initial sender uses higher layer for reliable data transfer (e.g., TCP), otherwise dropped data lost
- Ethernet's MAC protocol: Unslotted **CSMA/CD with binary backoff** <- *not so important anymore, because we have switches* ☺

# Time for a break

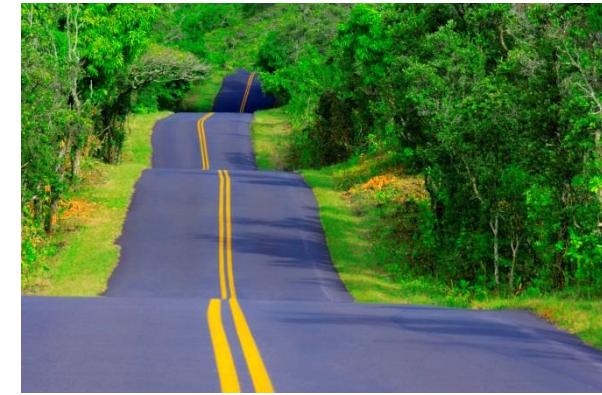
---



**Take a break, maybe a coffee or tea, and continue when you are back.**

# Roadmap Data-Link Layer

- Introduction to the link layer
- Error-Detection, Error-Correction
- Multiple access protocols (MAC)
- **Local Area Networks (LANs)**
  - Addressing, Address resolution protocol (ARP)
  - Ethernet
  - Switches
- Virtual Networks (VLANs)
- Data Centre Networking



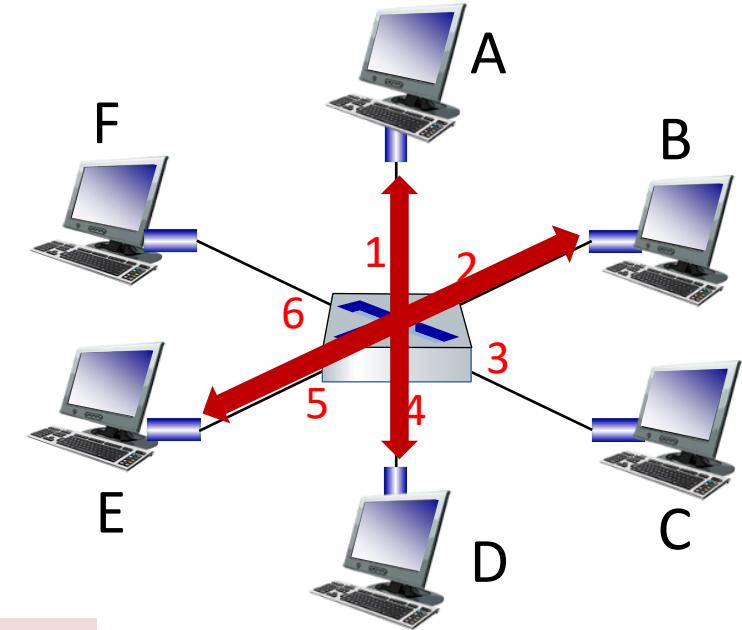
# Ethernet switch

---

- Switch is a **link-layer** device: takes an *active* role
  - Store and forward Ethernet frames
  - Examine incoming frame's MAC address, **selectively** forward frame to one-or-more outgoing links when frame is to be forwarded on segment
- **Transparent:** hosts unaware of presence of switches
- **plug-and-play, self-learning**
  - switches do not need to be configured typically (exceptions exists, see later!)

# Switch: Multiple simultaneous transmissions

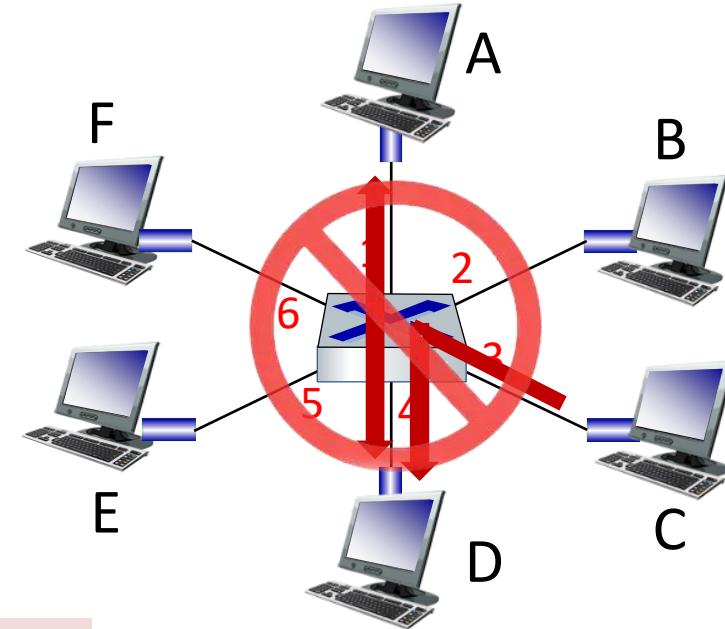
- Hosts have dedicated, direct connection to switch
- Switches buffer packets
- Ethernet protocol used on *each* incoming link, so:
  - **no collisions**; full duplex
- **switching**: A-to-D and B-to-E can transmit simultaneously, without collisions



switch with six  
interfaces (1,2,3,4,5,6)

# Switch: Multiple simultaneous transmissions

- Hosts have dedicated, direct connection to switch
- Switches buffer packets
- Ethernet protocol used on *each* incoming link, so:
  - **no collisions**; full duplex
- **switching**: A-to-D and B-to-E can transmit simultaneously, without collisions
  - but A-to-D and C to D can *not* happen simultaneously



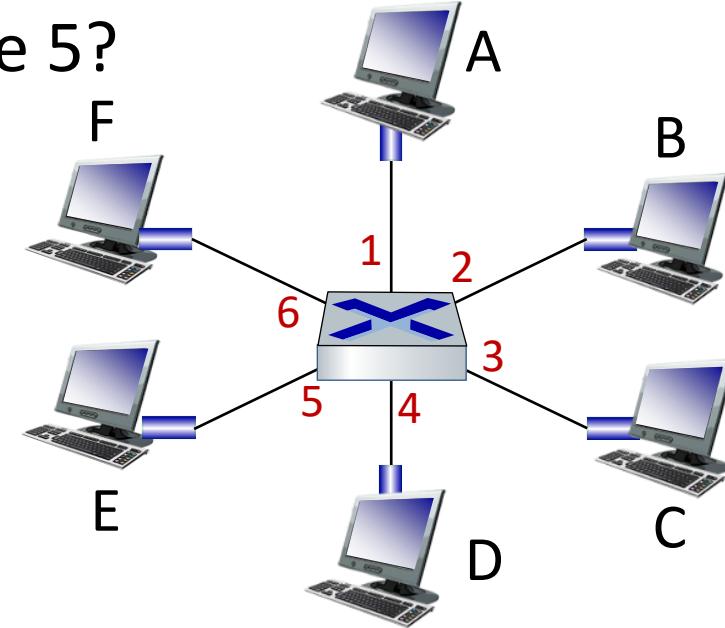
switch with six  
interfaces (1,2,3,4,5,6)

# Switch forwarding table

**Q:** how does switch know D is reachable via interface 4 and E is reachable via interface 5?

**A:** each switch has a **switch table**:

- (MAC address of host, interface to reach host, time stamp)
- looks like a routing table!

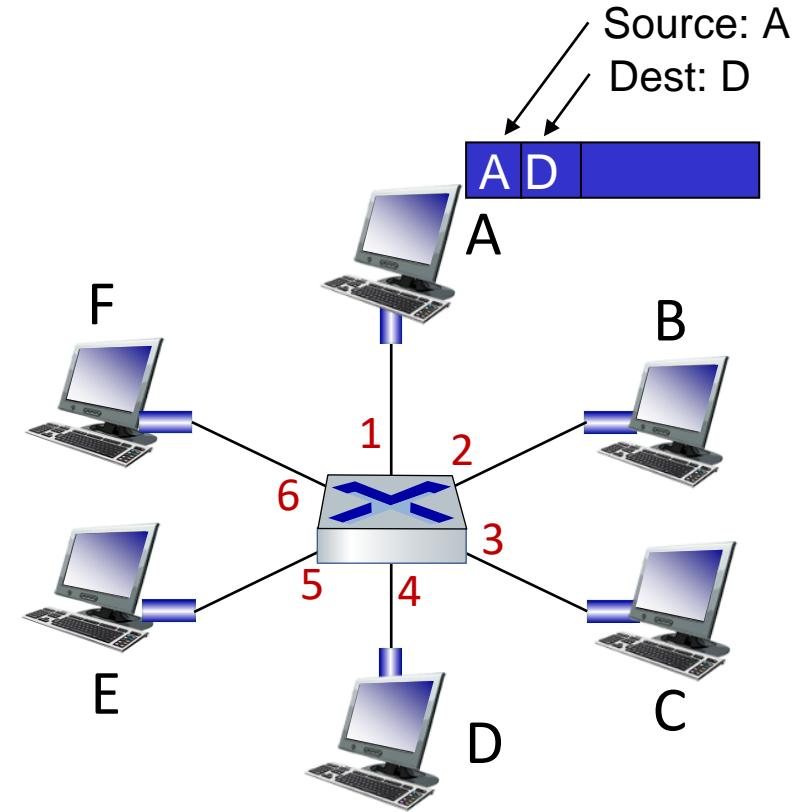


**Q:** how are entries created, maintained in switch table?

- self-learning

# Switch: self-learning

- Switch *learns* which hosts can be reached through which interfaces
  - When frame received, switch “learns” location of sender: incoming LAN segment
  - Records sender/location pair in switch table



| MAC addr | interface | TTL |
|----------|-----------|-----|
| A        | 1         | 60  |

*Switch table  
(initially empty)*

# Switch: frame filtering/forwarding

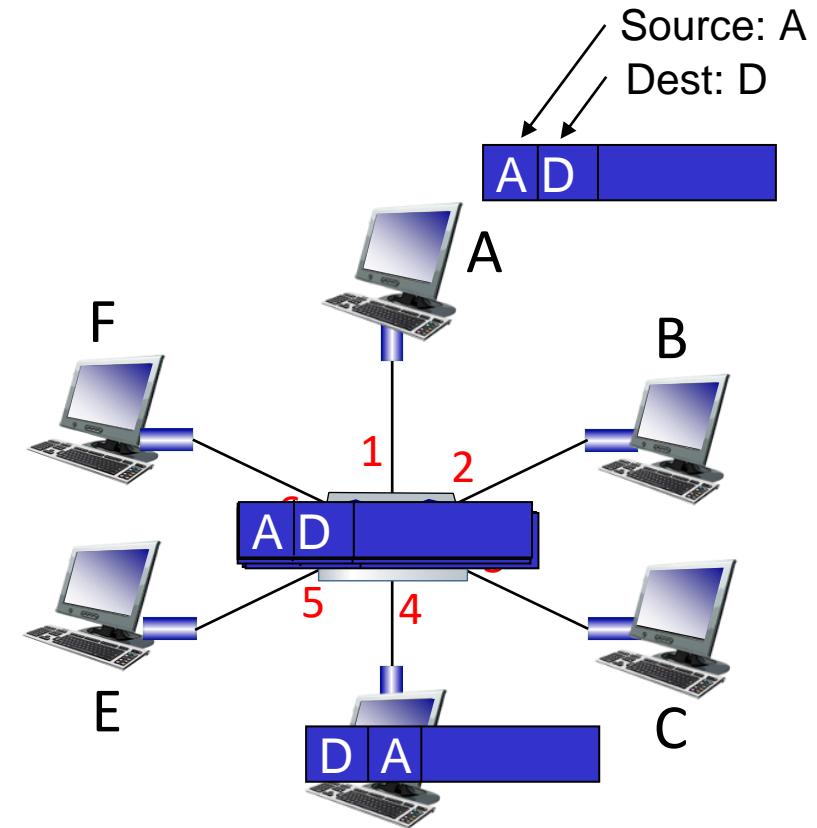
---

when frame received at switch:

1. record incoming link, MAC address of sending host
2. index switch table using MAC destination address
3. if entry found for destination
  - then {
    - if destination on segment from which frame arrived
      - then drop frame
      - else forward frame on interface indicated by entry
  - }
  - else flood /\* forward on all interfaces except arriving interface \*/

# Self-learning, forwarding: An example

- Frame destination, D, location unknown: **flood**
- Destination A location known: **selectively send on just one link**

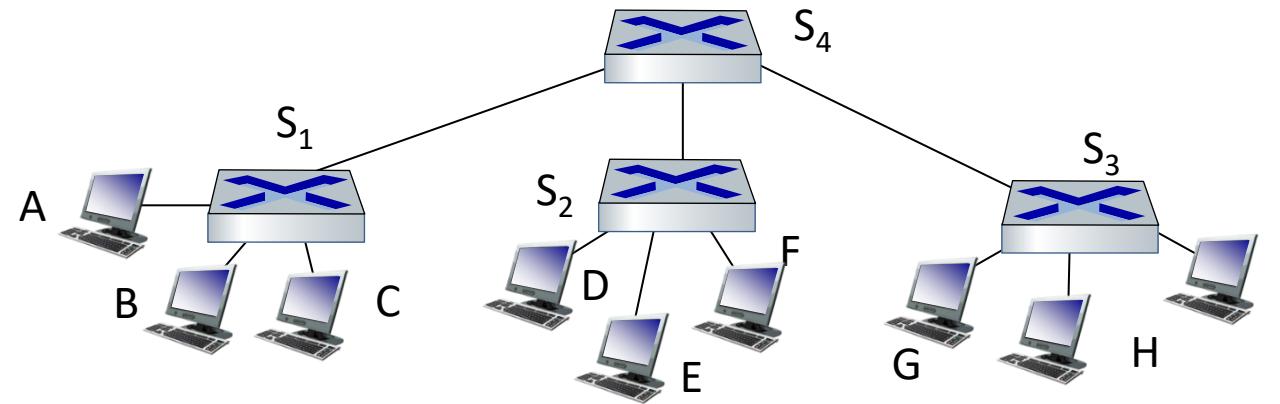


| MAC addr | interface | TTL |
|----------|-----------|-----|
| A        | 1         | 60  |
| D        | 4         | 60  |

*switch table  
(initially empty)*

# Interconnecting switches

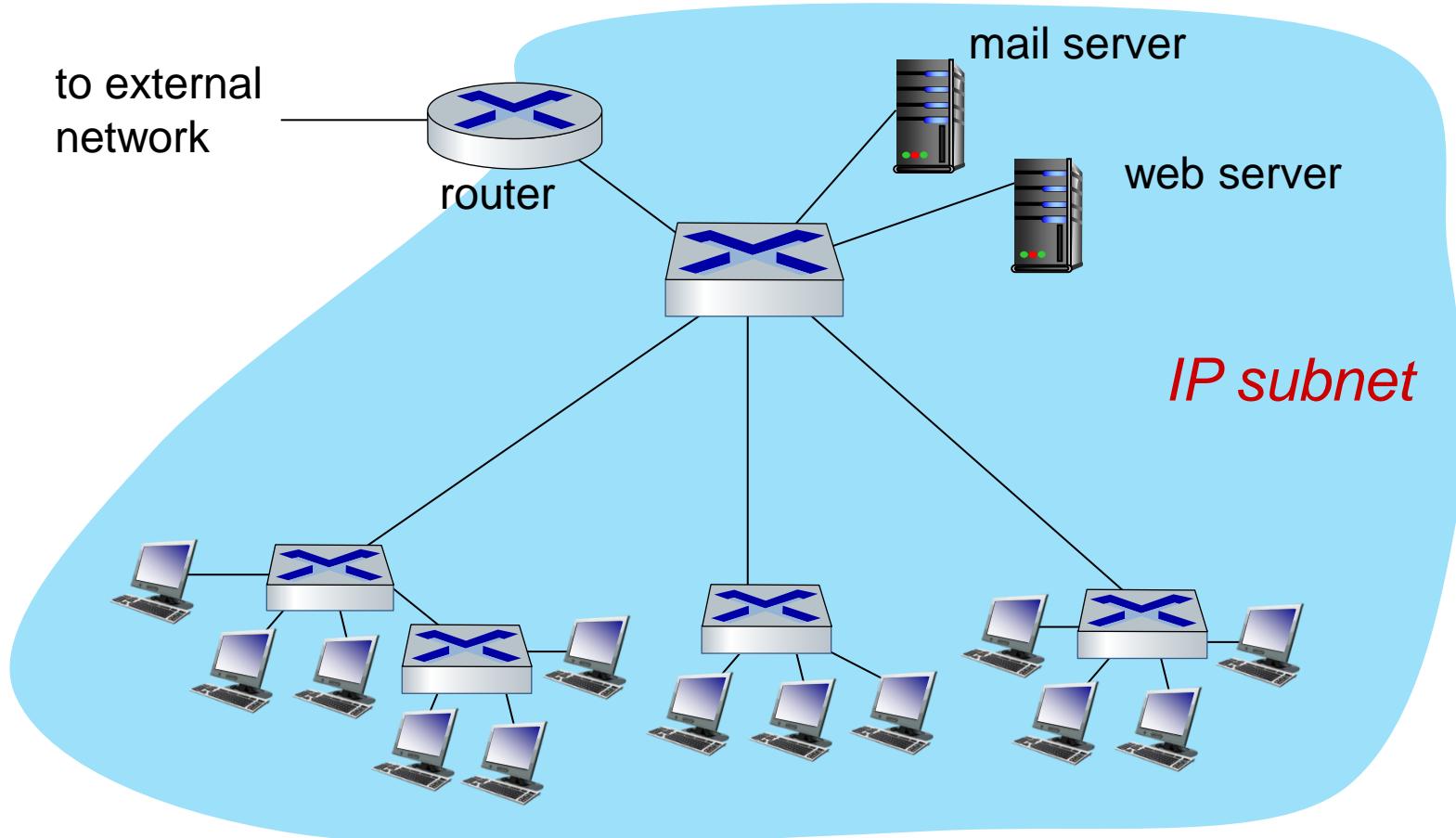
Self-learning switches can be connected together:



**Q:** Sending from A to G - how does  $S_1$  know to forward frame destined to G via  $S_4$  and  $S_3$ ?

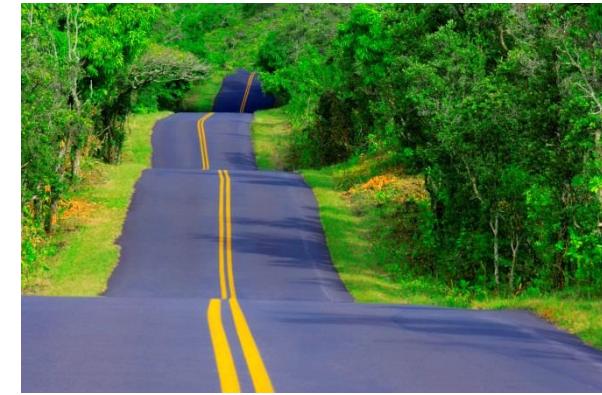
- **A:** Self learning! (works exactly the same as in single-switch case!)

# Small institutional network



# Roadmap Data-Link Layer

- Introduction to the link layer
- Error-Detection, Error-Correction
- Multiple access protocols (MAC)
- Local Area Networks (LANs)
  - Addressing, Address resolution protocol (ARP)
  - Ethernet
  - Switches
- **Virtual Networks (VLANs)**
- Data Centre Networking

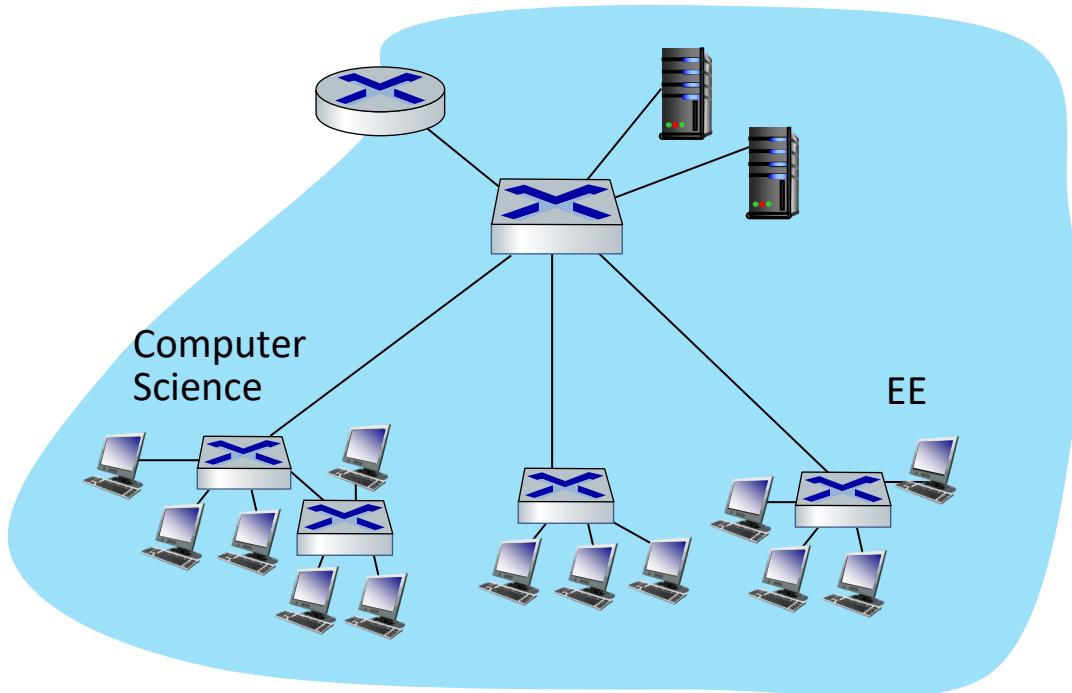


# Virtual LANs (VLANs): Motivation

*Q:* What happens as LAN sizes scale, users change point of attachment?

Single broadcast domain:

- *scaling:* all layer-2 broadcast traffic (ARP, DHCP, unknown MAC) must cross entire LAN
- efficiency, security, privacy, efficiency issues

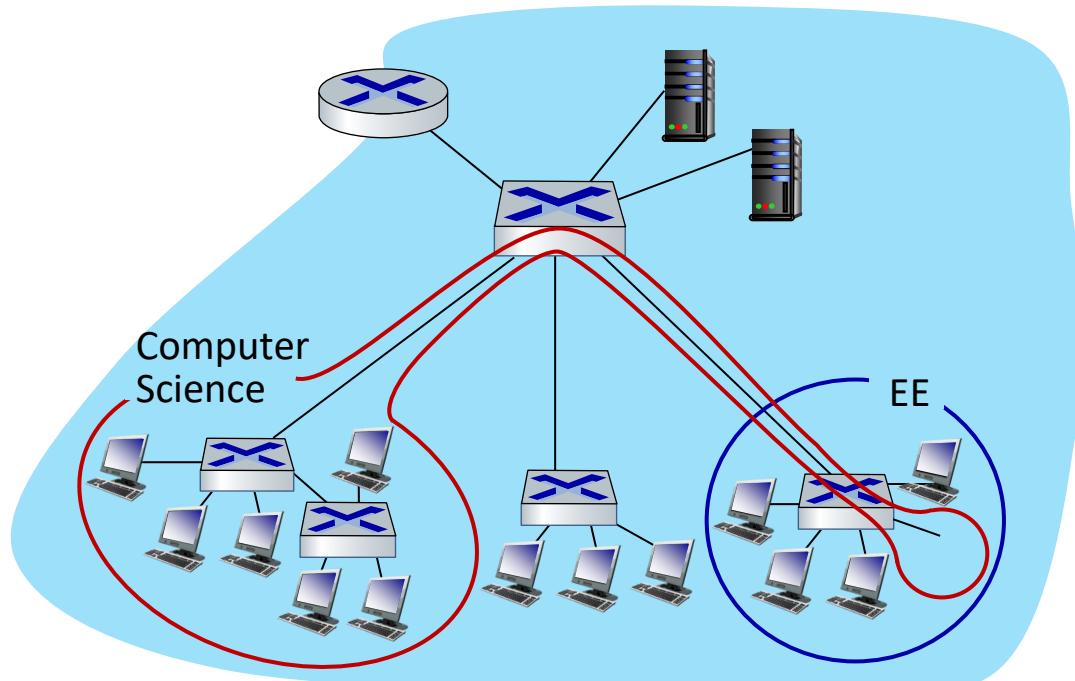


# Virtual LANs (VLANs): Motivation

Q: What happens as LAN sizes scale, users change point of attachment?

Single broadcast domain:

- *scaling*: all layer-2 broadcast traffic (ARP, DHCP, unknown MAC) must cross entire LAN
- efficiency, security, privacy, efficiency issues



Administrative issues:

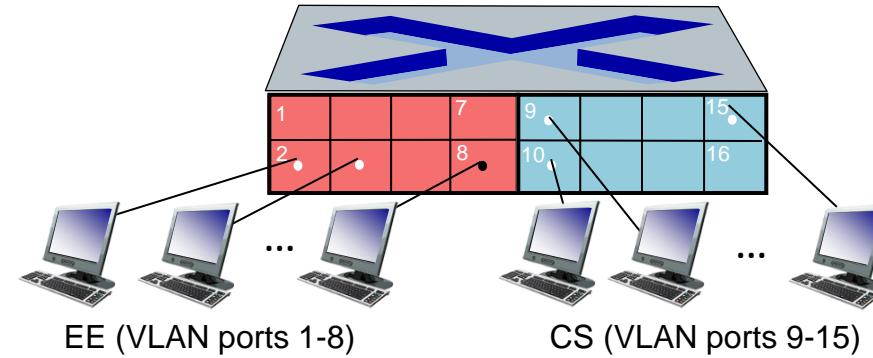
- CS user moves office to EE - *physically* attached to EE switch, but wants to remain *logically* attached to CS switch

# Port-based VLANs

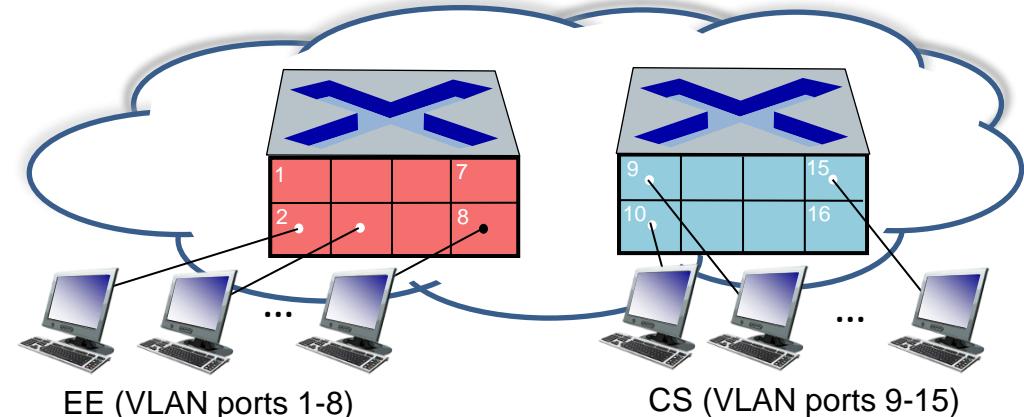
## Virtual Local Area Network (VLAN)

Switch(es) supporting VLAN capabilities can be configured to define **multiple *virtual* LANs** over single physical LAN infrastructure.

**port-based VLAN:** switch ports grouped (by switch management software) so that *single* physical switch .....

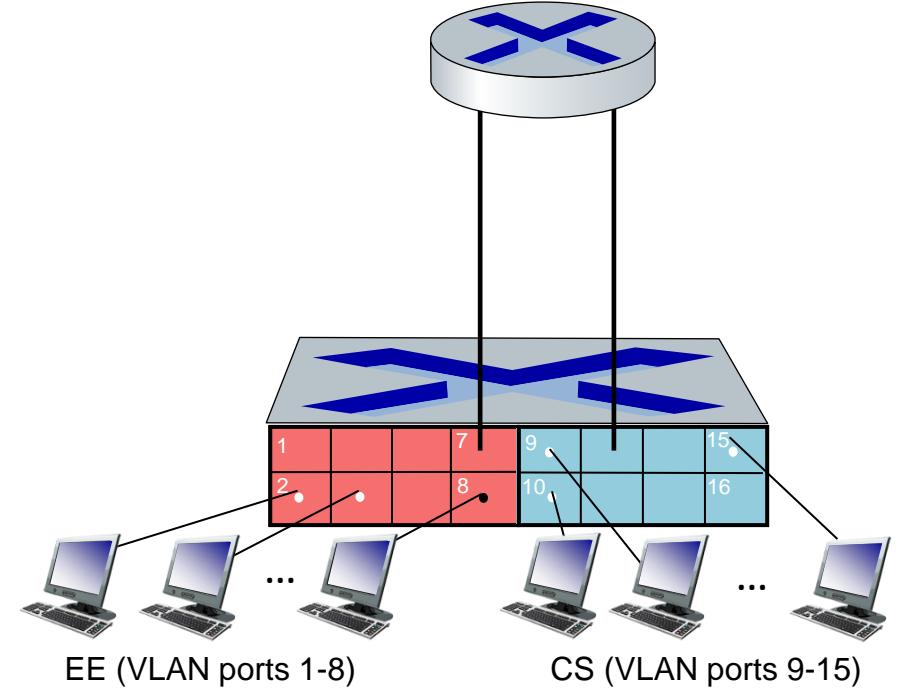


... operates as **multiple virtual switches**

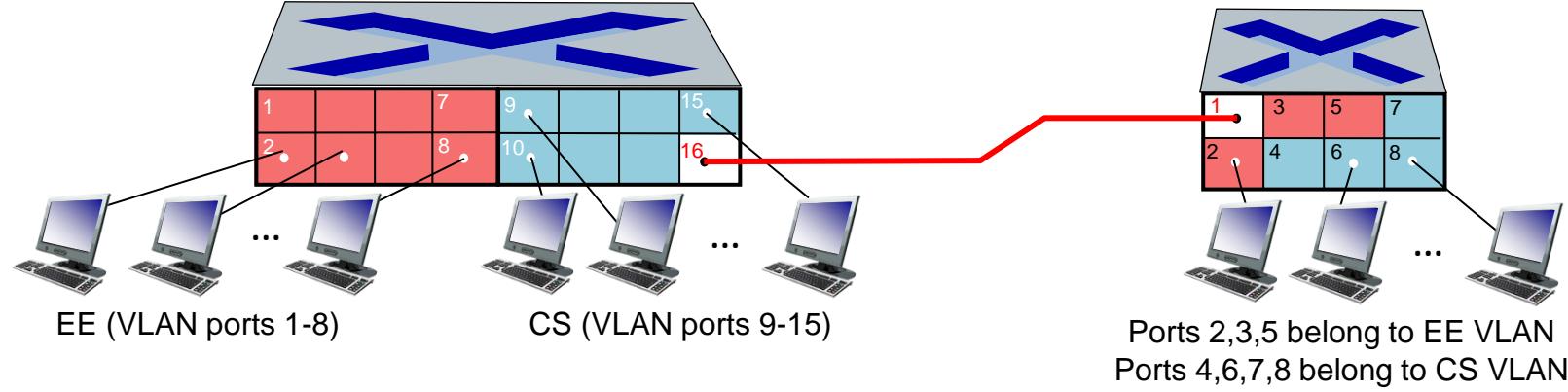


# Port-based VLANs

- **Traffic isolation:** frames to/from ports 1-8 can *only* reach ports 1-8
  - can also define VLAN based on MAC addresses of endpoints, rather than switch port
- **Dynamic membership:** ports can be dynamically assigned among VLANs
- **Forwarding between VLANs:** done via routing (just as with separate switches)
  - in practice vendors sell combined switches plus routers



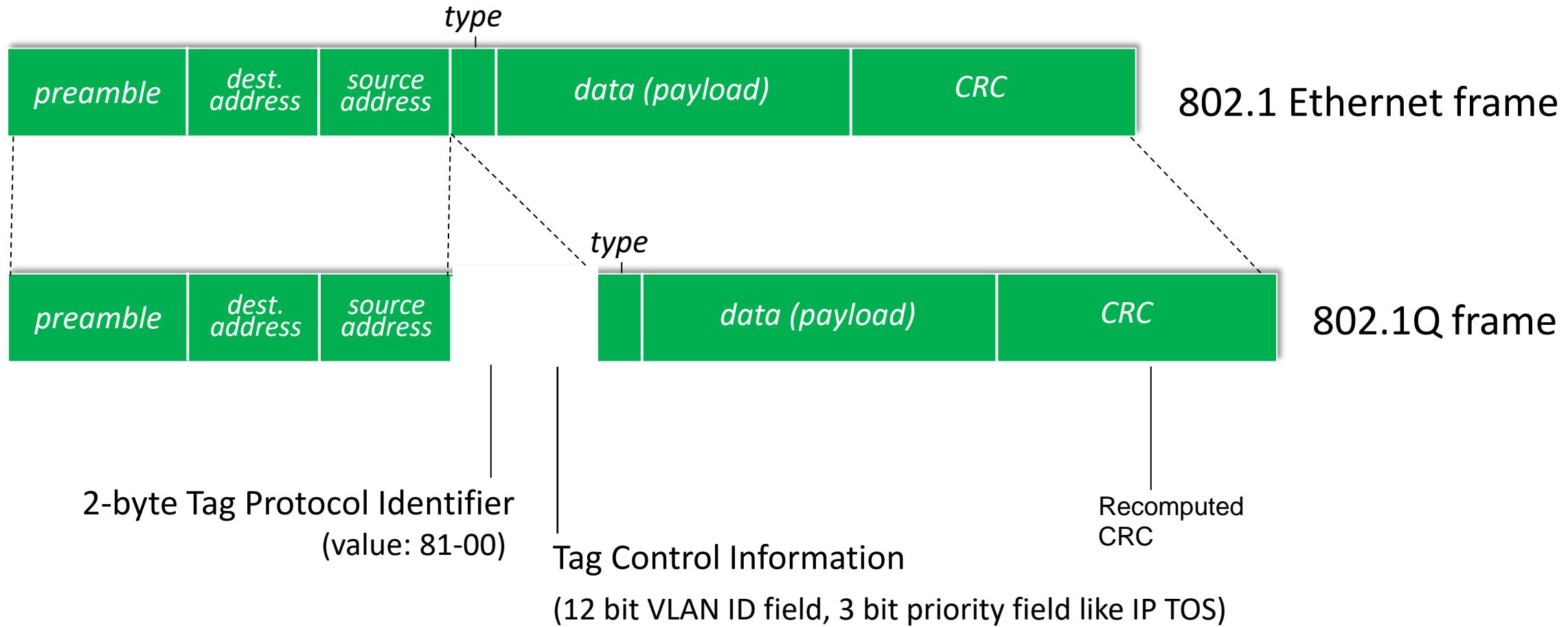
# VLANs spanning multiple switches



**Trunk port:** carries frames between VLANs defined over multiple physical switches

- frames forwarded within VLAN between switches can't be vanilla 802.1 frames (must carry VLAN ID info)
- 802.1q protocol adds/removed additional header fields for frames forwarded between trunk ports

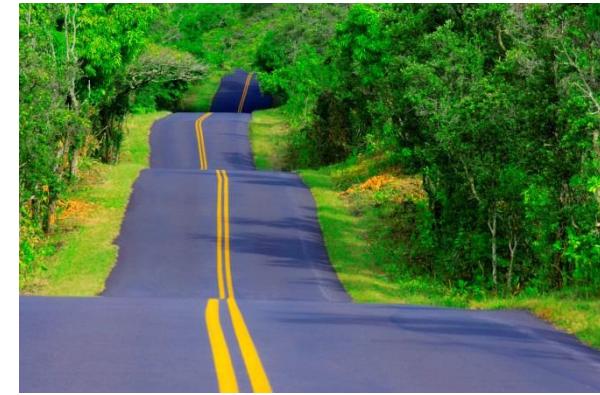
# 802.1Q VLAN frame format



# Roadmap Data-Link Layer



- Introduction to the link layer
- Error-Detection, Error-Correction
- Multiple access protocols (MAC)
- Local Area Networks (LANs)
  - Addressing, Address resolution protocol (ARP)
  - Ethernet
  - Switches
- Virtual Networks (VLANs)
- Data Centre Networking



# Datacenter networks

10's to 100's of thousands of hosts, often closely coupled, in close proximity:

- E-business (e.g. Amazon)
- Content-servers (e.g., YouTube, Akamai, Apple, Microsoft)
- Search engines, data mining (e.g., Google)

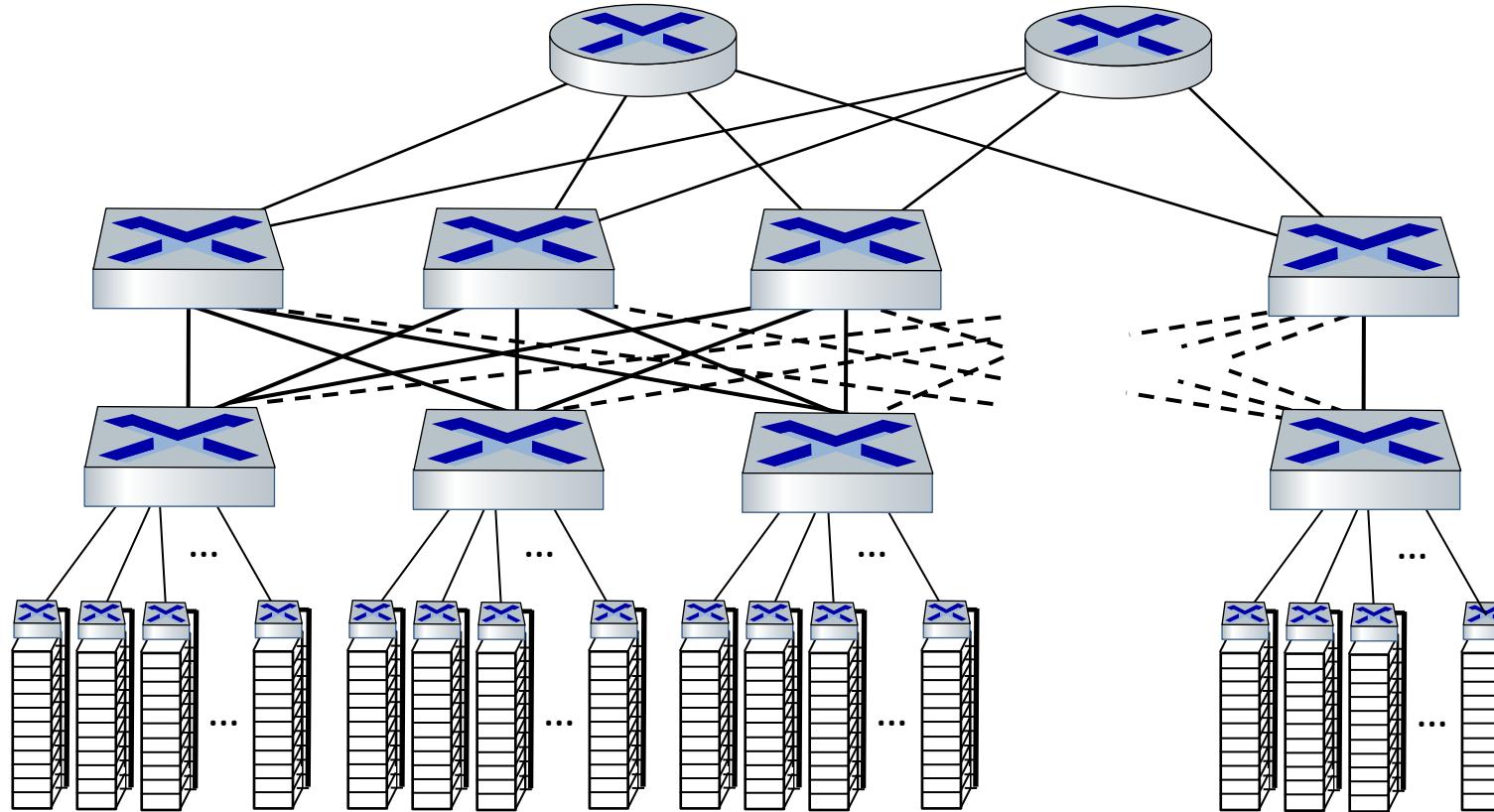
Challenges:

- Multiple applications, each serving massive numbers of clients
- Reliability
- Managing/balancing load, avoiding processing, networking, data bottlenecks



Inside a Microsoft container, Chicago data center

# Datacenter networks: Network elements



## Border routers

- connections outside datacenter

## Tier-1 switches

- connecting to ~16 T-2s below

## Tier-2 switches

- connecting to ~16 TORs below

## Top of Rack (TOR) switch

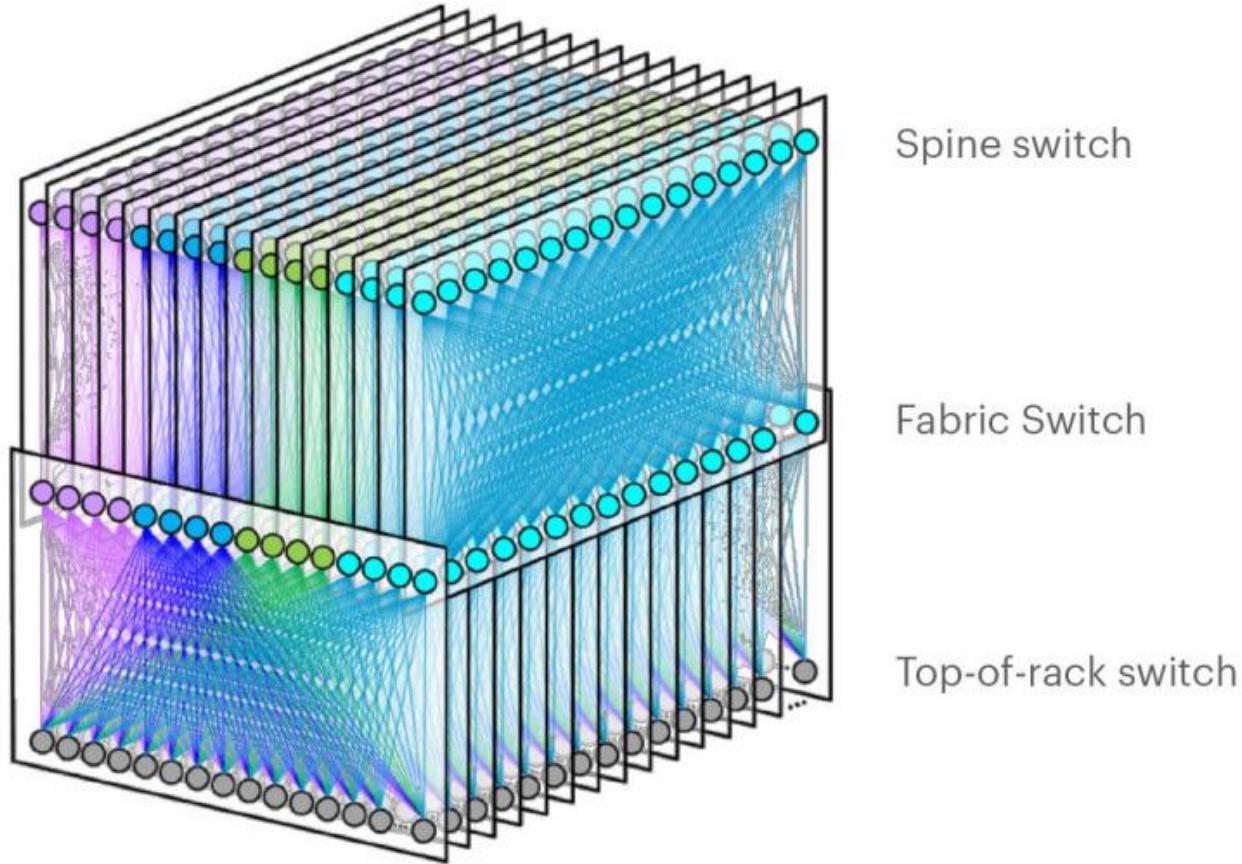
- one per rack
- 40-100Gbps Ethernet to blades

## Server racks

- 20- 40 server blades: hosts

# Datacenter networks: network elements

Facebook F16 data center network topology:



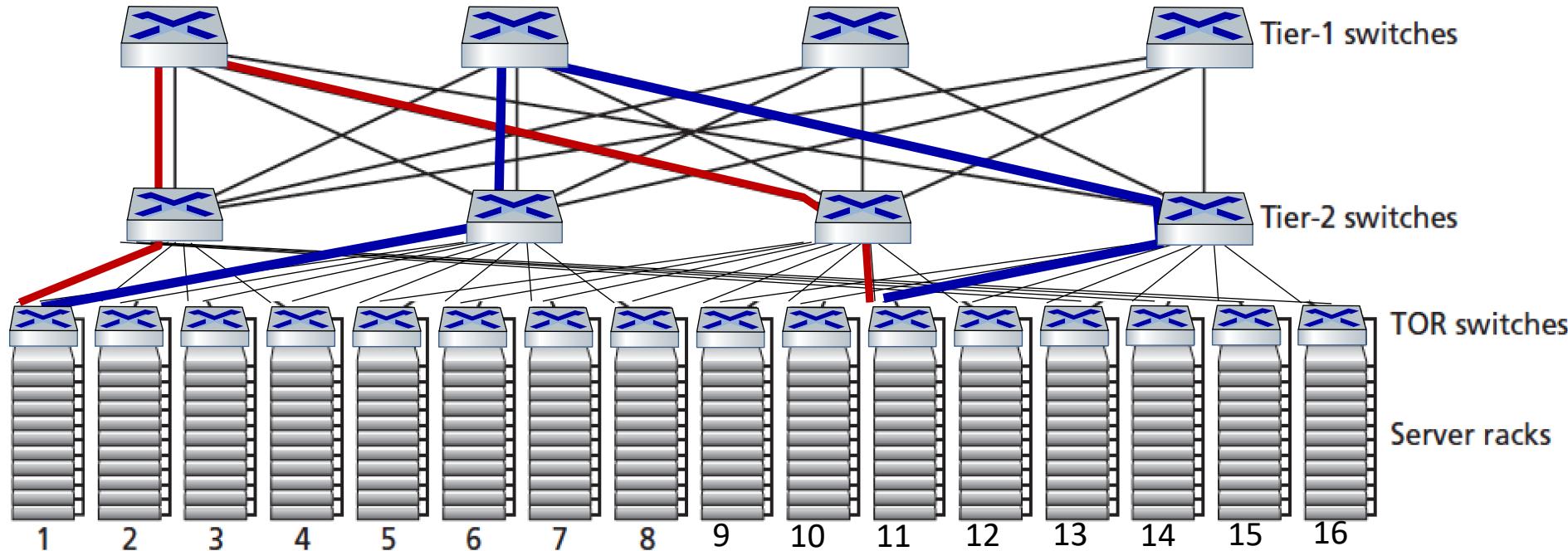
<https://engineering.fb.com/data-center-engineering/f16-minipack/> (posted 3/2019)



© Meta Datacenter | Luleå, Nya Teknik

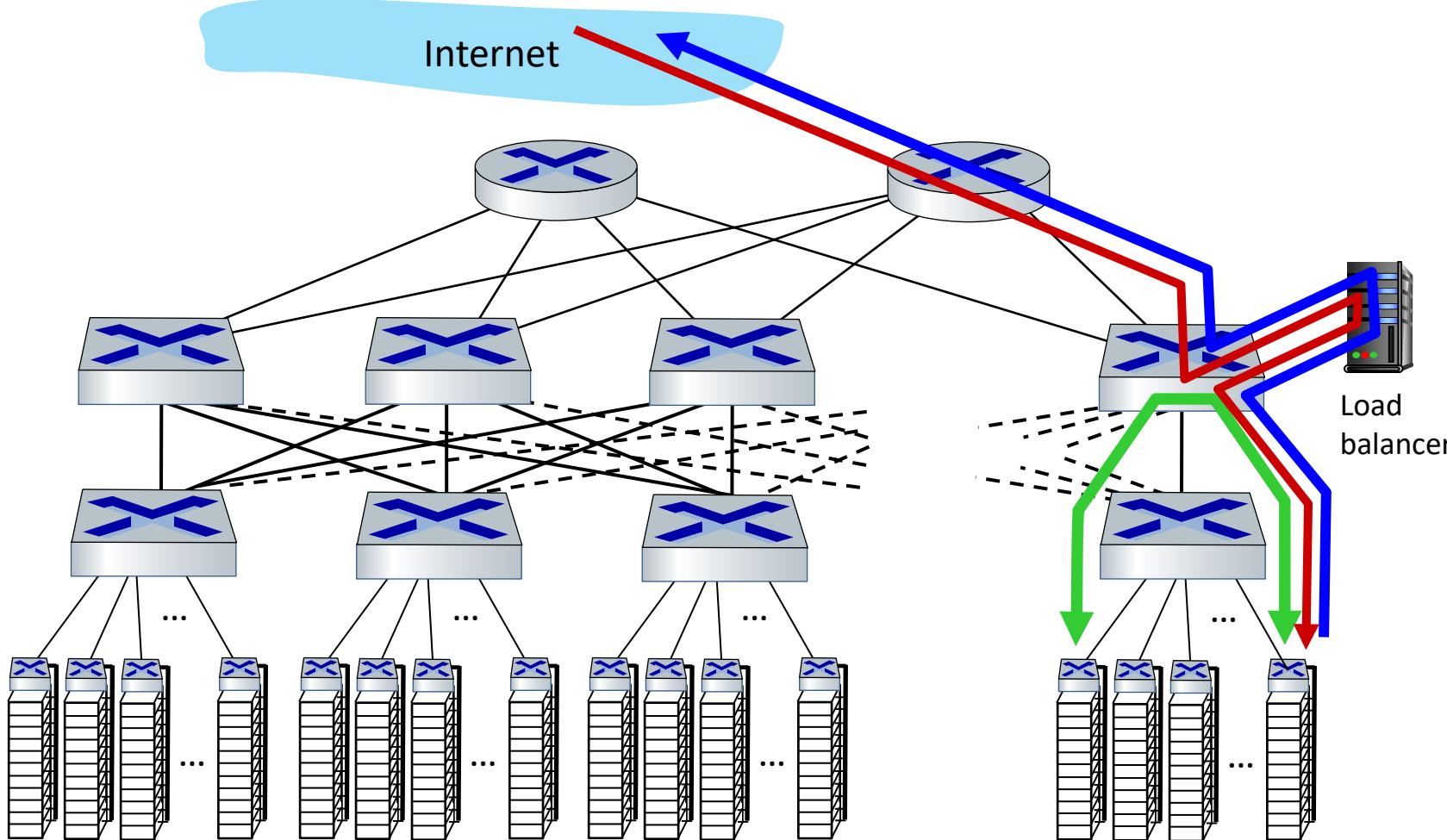
# Datacenter networks: Multipath

- Rich interconnection among switches, racks:
  - Increased throughput between racks (multiple routing paths possible)
  - Increased reliability via redundancy



two **disjoint** paths highlighted between racks 1 and 11

# Datacenter networks: Application-layer routing



## Load balancer: application-layer routing

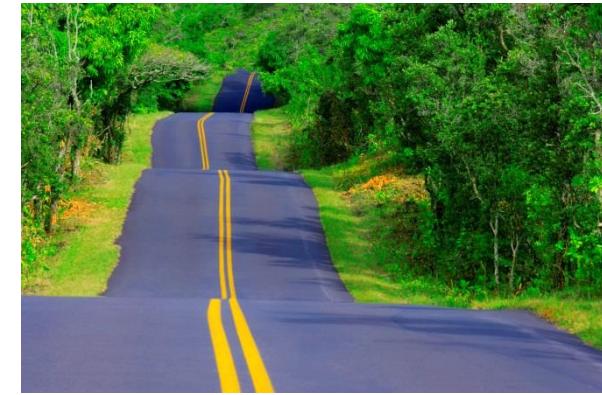
- receives external client requests
- directs workload within data center
- returns results to external client (hiding data center internals from client)

# Datacenter networks: Protocol innovations

- **Link layer:**
  - RoCE: remote DMA (Direct Memory Access) over Converged Ethernet
- **Transport layer:**
  - ECN (explicit congestion notification) used in transport-layer congestion control (DCTCP, DCQCN)
  - Experimentation with hop-by-hop (backpressure) congestion control
- **Routing, management:**
  - Software-defined-networking (SDN) widely used within/among organizations' datacenters
  - Place related services, data as close as possible (e.g., in same rack or nearby rack) to minimize tier-2, tier-1 communication

# Roadmap Data-Link Layer

- Introduction to the link layer
- Error-Detection, Error-Correction
- Multiple access protocols (MAC)
- Local Area Networks (LANs)
  - Addressing, Address resolution protocol (ARP)
  - Ethernet
  - Switches
- Virtual Networks (VLANs)
- Data Centre Networking



# Chapter 6: Summary

---

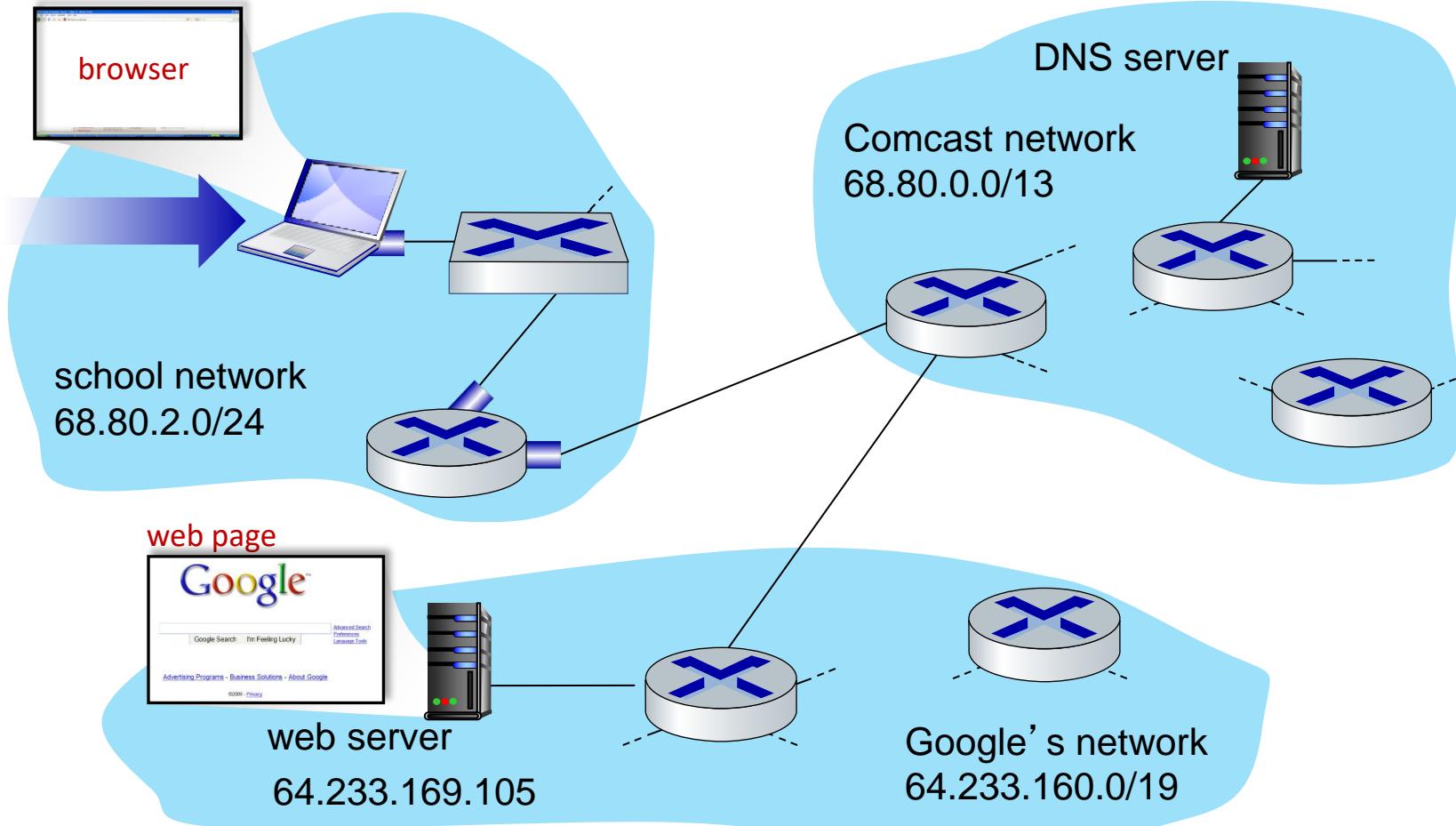
- Principles behind data link layer services:
  - Error detection, correction
  - Sharing a broadcast channel: multiple access
  - Link layer addressing
- Instantiation, implementation of various link layer technologies
  - Ethernet
  - Switched LANs
  - Virtualized networks, VLANs
- Synthesis: A day in the life of a web request

# A day in the life of a web request

---

- Putting-it-all-together: A day in the life of a web request.
  - *Goal:* identify, review, understand protocols (at all layers) involved in seemingly simple scenario: requesting www page
  - *Scenario:* student attaches laptop to campus network, requests/receives www.google.com

# A day in the life of a web request: The scenario

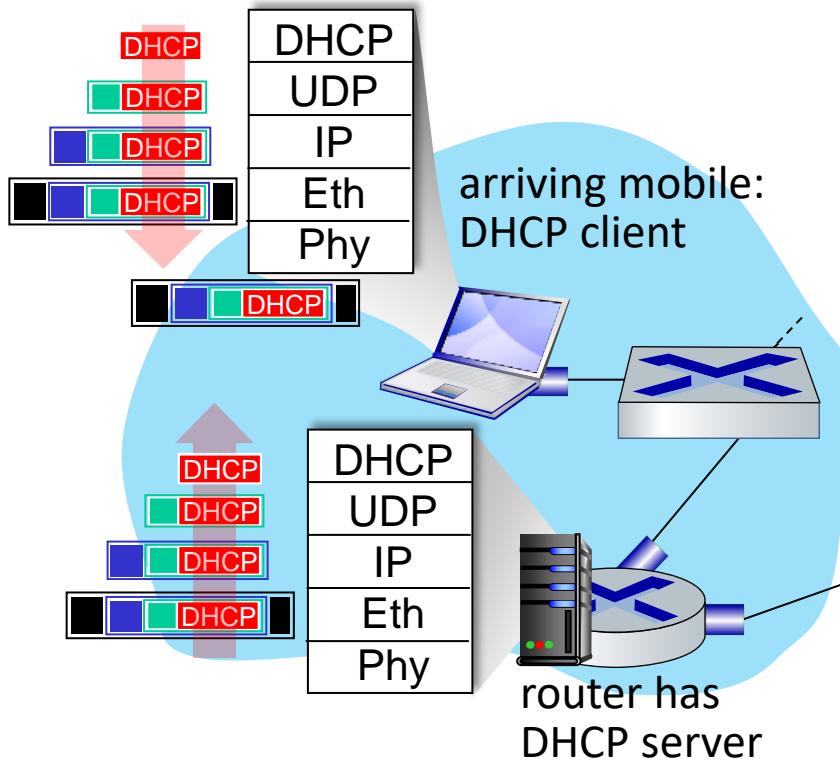


scenario:

- arriving mobile client attaches to network ...
- requests web page:  
www.google.com

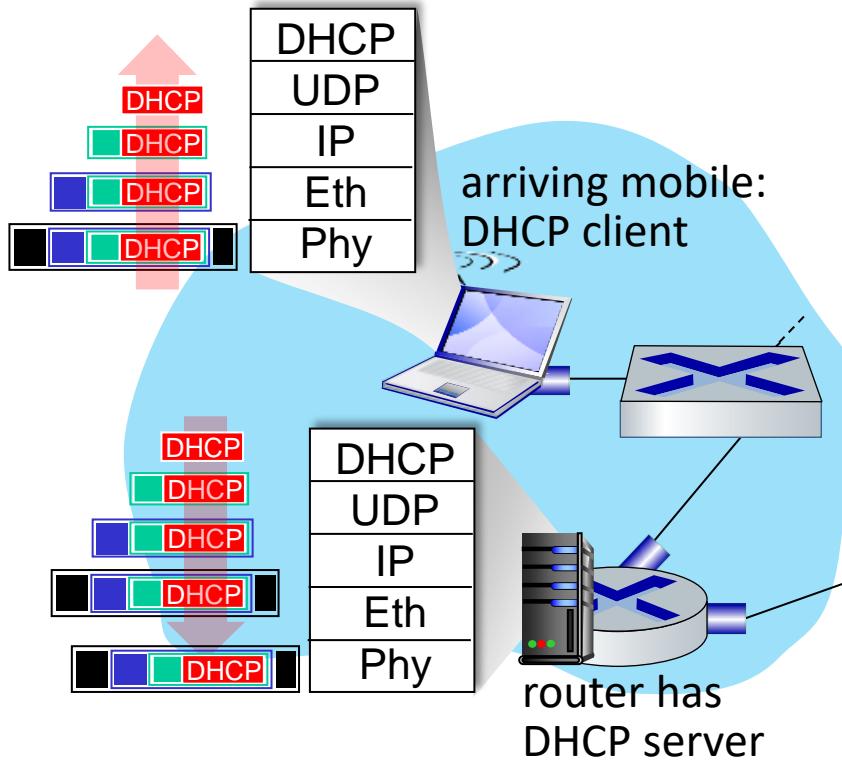
*Sounds simple!* !

# A day in the life: connecting to the Internet



- connecting laptop needs to get its own IP address, addr of first-hop router, addr of DNS server: use **DHCP**
- DHCP request encapsulated in **UDP**, encapsulated in **IP**, encapsulated in **802.3** Ethernet frame
- Ethernet frame **broadcast** (dest: FFFFFFFFFFFF) on LAN, received at router running **DHCP** server
- Ethernet **demuxed** to IP, demuxed to UDP, demuxed to DHCP

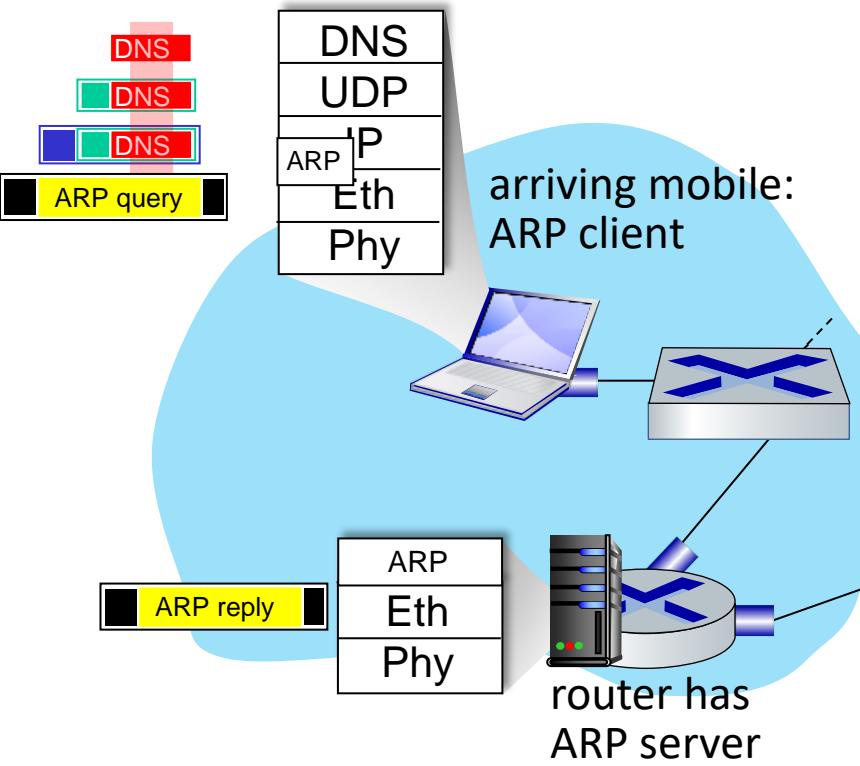
# A day in the life: connecting to the Internet



- DHCP server formulates **DHCP ACK** containing client's IP address, IP address of first-hop router for client, name & IP address of DNS server
- encapsulation at DHCP server, frame forwarded (**switch learning**) through LAN, demultiplexing at client
- DHCP client receives DHCP ACK reply

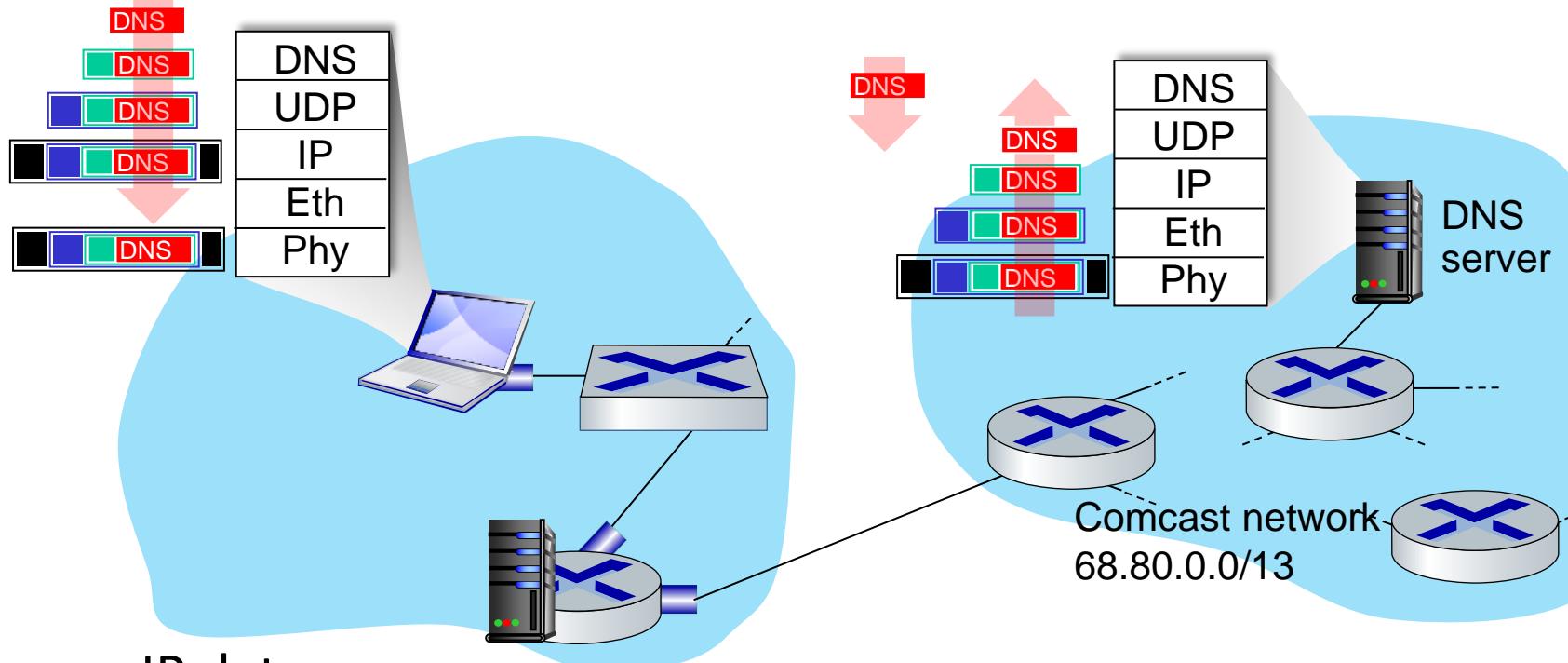
*Client now has IP address, knows name & addr of DNS server, IP address of its first-hop router*

# A day in the life... ARP (before DNS, before HTTP)



- before sending **HTTP** request, need IP address of [www.google.com](http://www.google.com): **DNS**
- DNS query created, encapsulated in UDP, encapsulated in IP, encapsulated in Ethernet frame. To send frame to router, need MAC address of router interface: **ARP**
- **ARP query** broadcast, received by router, which replies with **ARP reply** giving MAC address of router interface
- client now knows MAC address of first hop router, so can now send frame containing DNS query

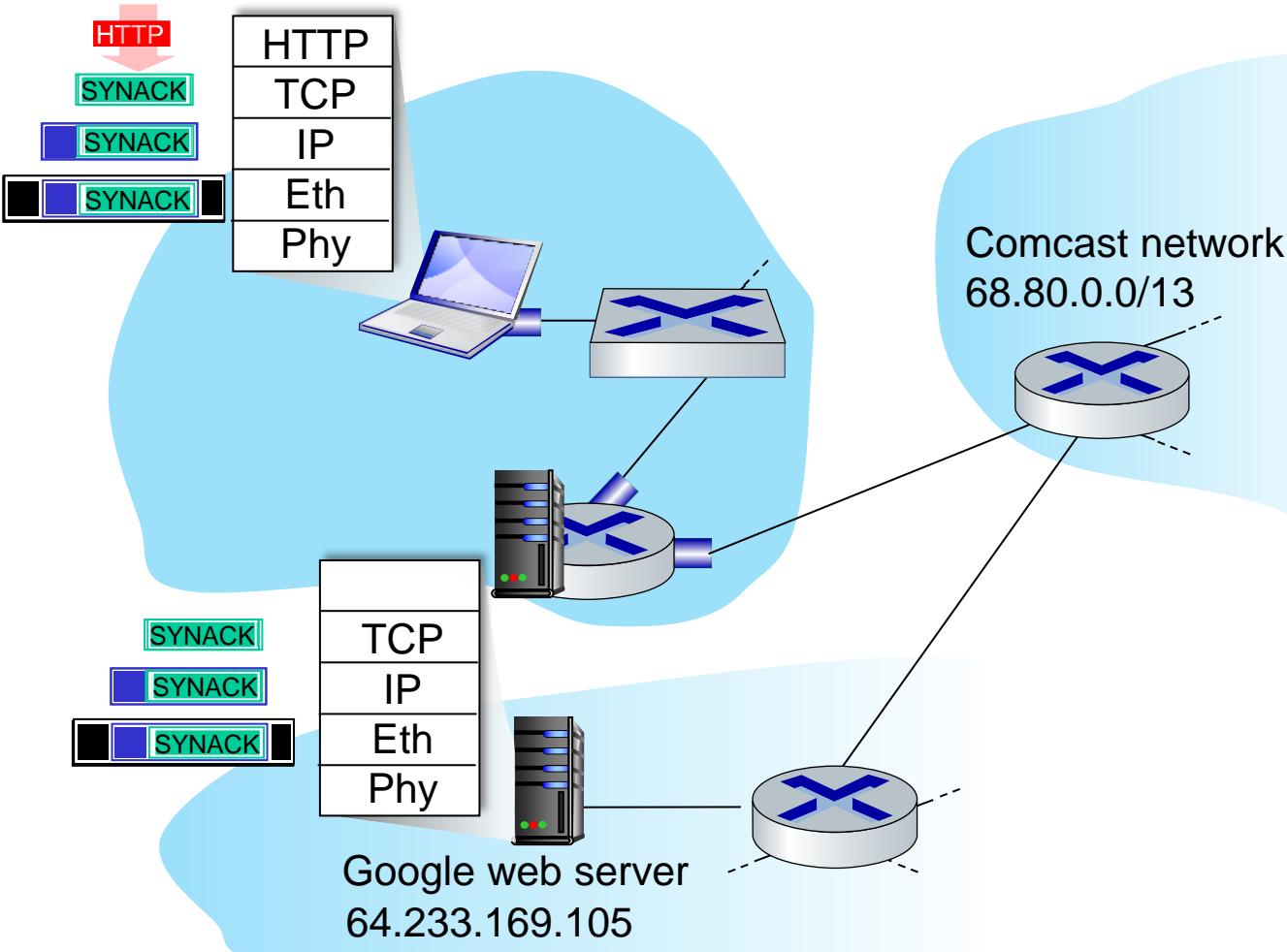
# A day in the life... using DNS



- IP datagram containing DNS query forwarded via LAN switch from client to 1<sup>st</sup> hop router
- IP datagram forwarded from campus network into Comcast network, routed (tables created by **RIP, OSPF, IS-IS** and/or **BGP** routing protocols) to DNS server

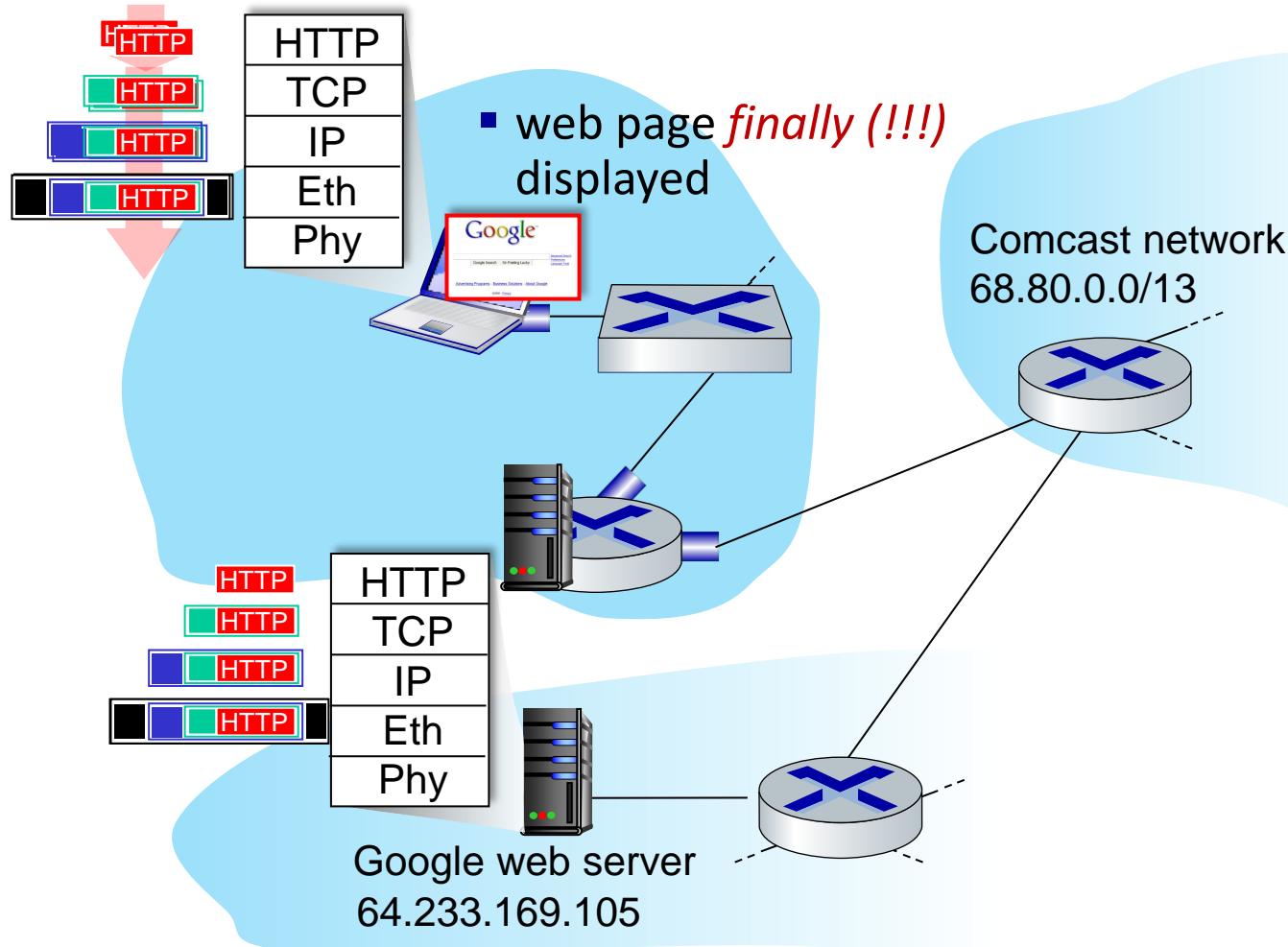
- demuxed to DNS
- DNS replies to client with IP address of [www.google.com](http://www.google.com)

# A day in the life...TCP connection carrying HTTP



- to send HTTP request, client first opens **TCP socket** to web server
- **TCP SYN segment** (step 1 in TCP 3-way handshake) inter-domain routed to web server
- web server responds with **TCP SYNACK** (step 2 in TCP 3-way handshake)
- **TCP connection established!**

# A day in the life... HTTP request/reply



- **HTTP request** sent into TCP socket
- IP datagram containing HTTP request routed to [www.google.com](http://www.google.com)
- web server responds with **HTTP reply** (containing web page)
- IP datagram containing HTTP reply routed back to client

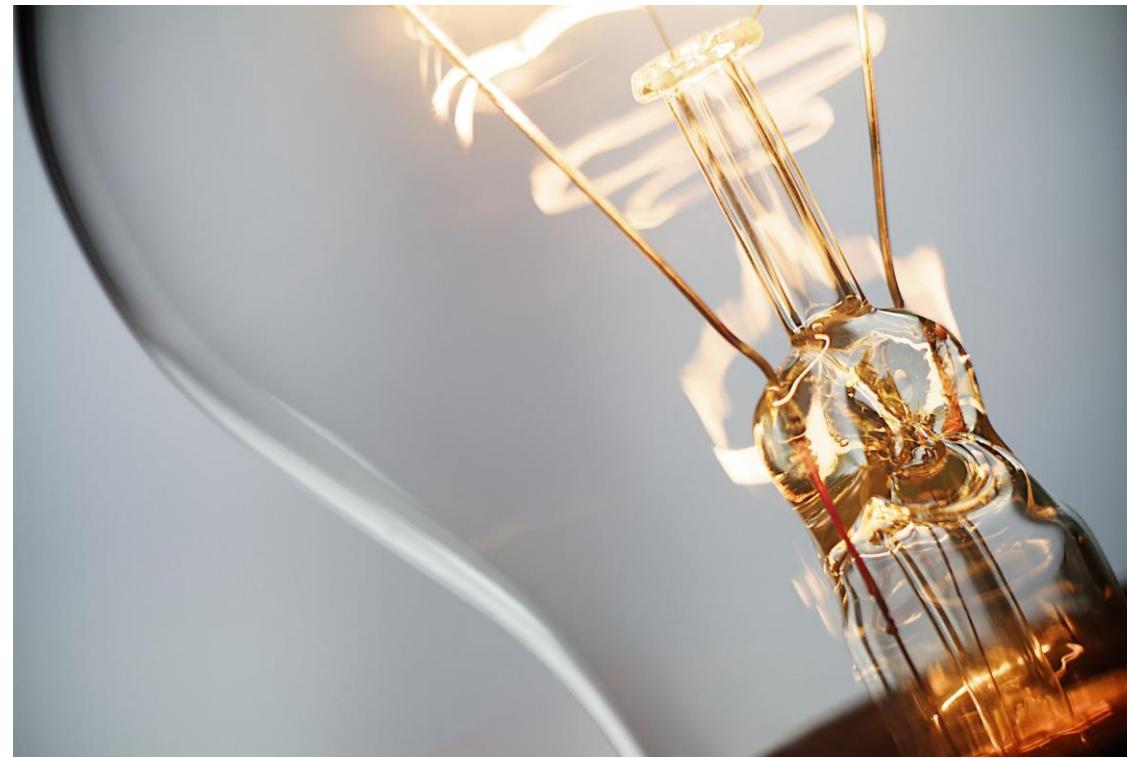
# What is coming next?



- Wireless Networking



- Security



- New Ideas: Evolving Networks

# Course on Computer Communication and Networks

## Lecture 10

### Continuously evolving Internet-working

Part A: Network edge

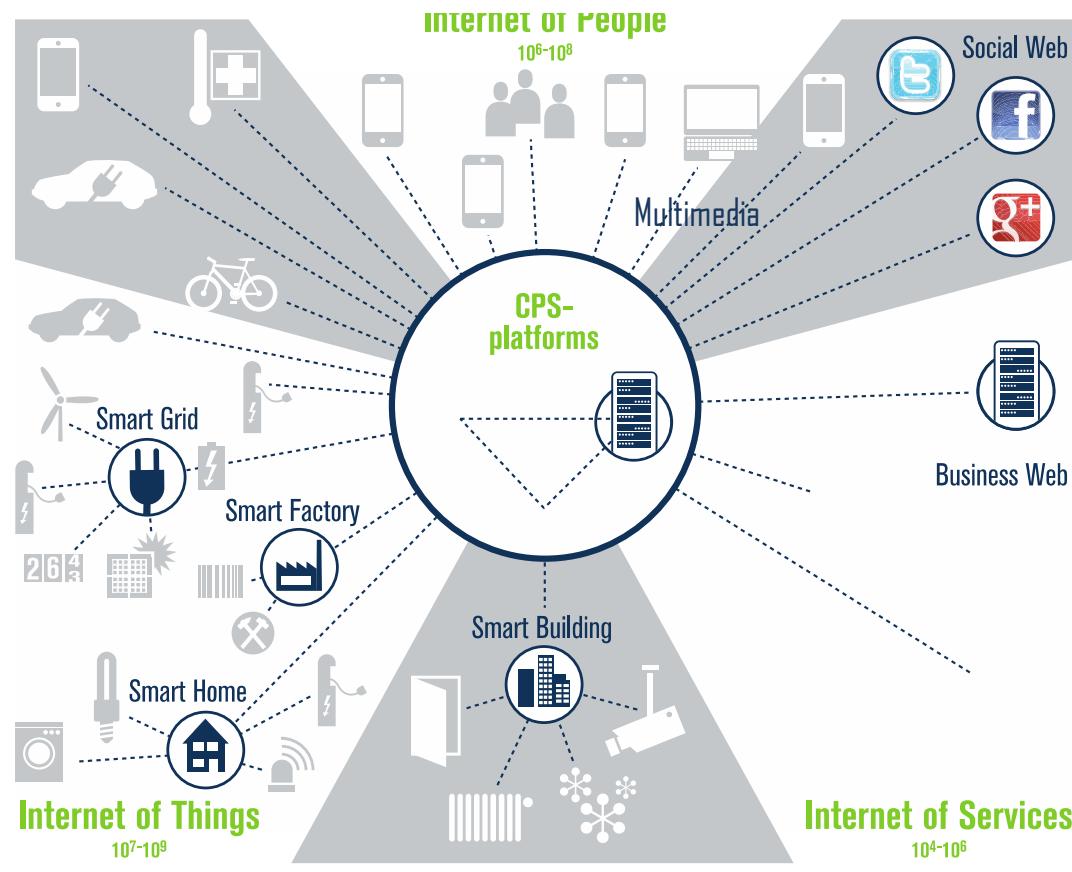
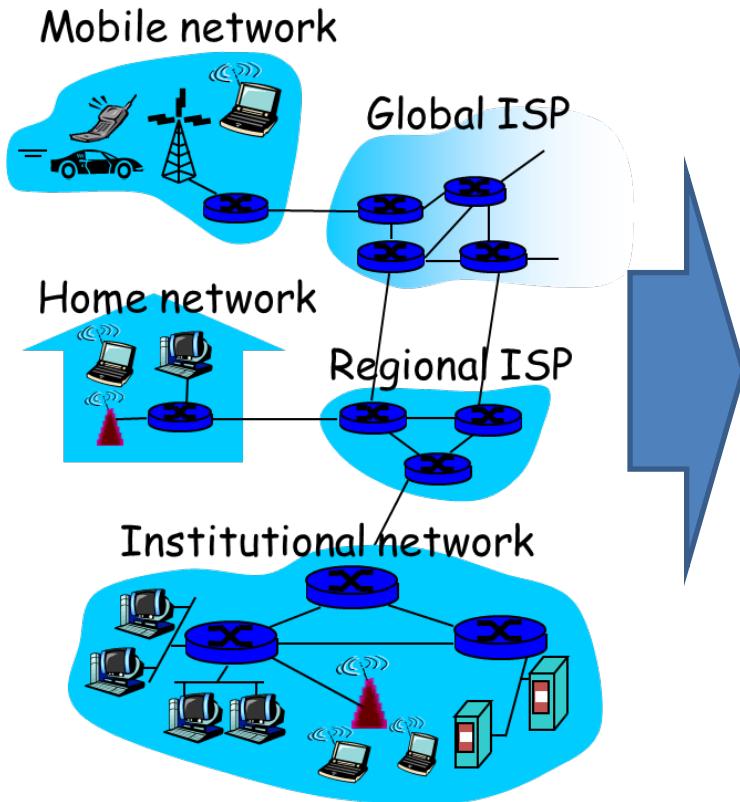
p2p NW, media streaming, http&transport-layer-updates, CDN

EDA344 / DIT 423 / LEU062 CTH/GU

Lecturer Marina Papatriantafilou

Based on the book Computer Networking: A Top Down Approach, Jim Kurose, Keith Ross, Addison-Wesley.

# Internet & its context....

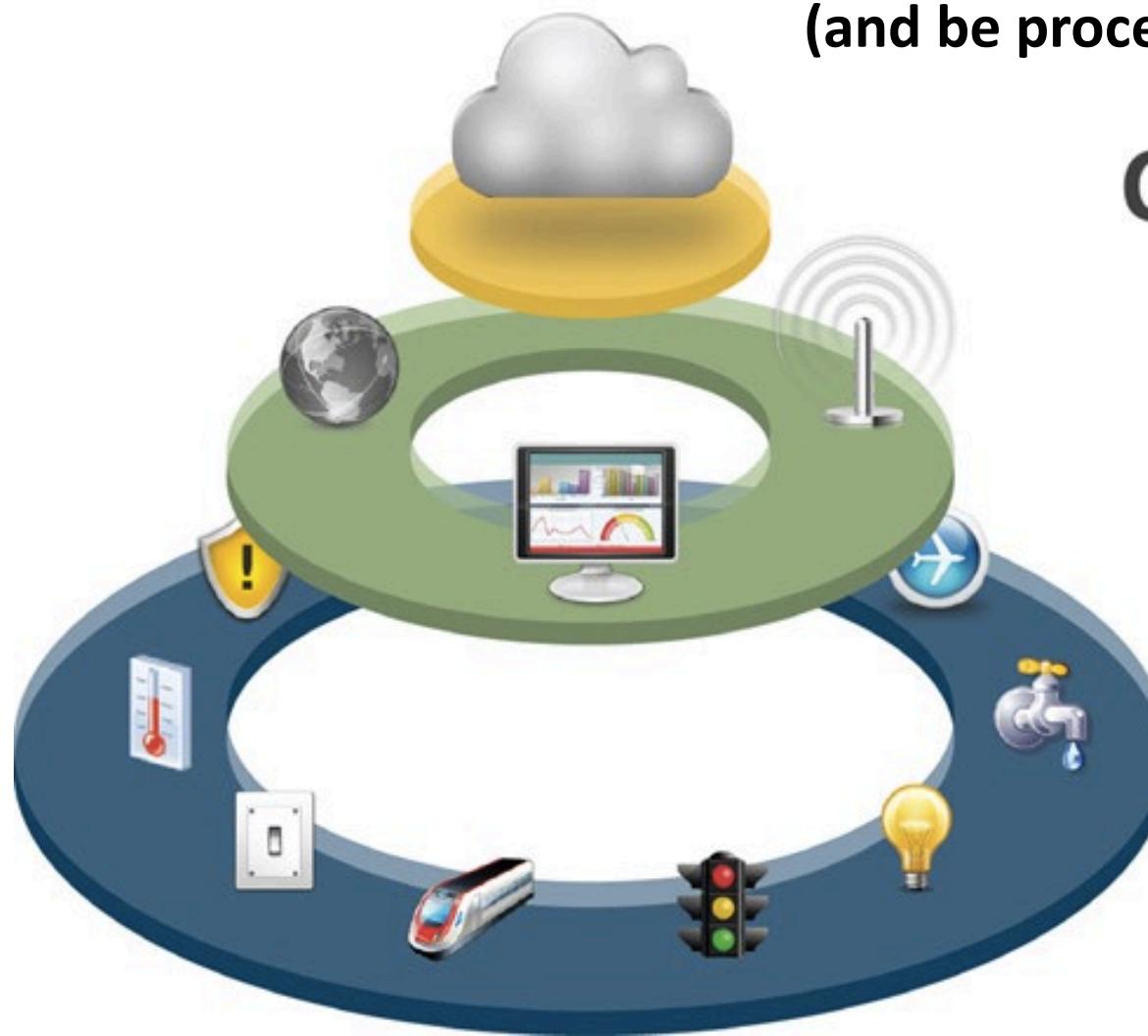


Source: Bosch Software Innovations 2012

continuous evolution ....

# "Internet-of-X" - Data processing/ML - Distributed Computing in interplay

**Evolving needs:** A lot of data to be communicated  
(and be processed)



<http://www.iebmedia.com/>

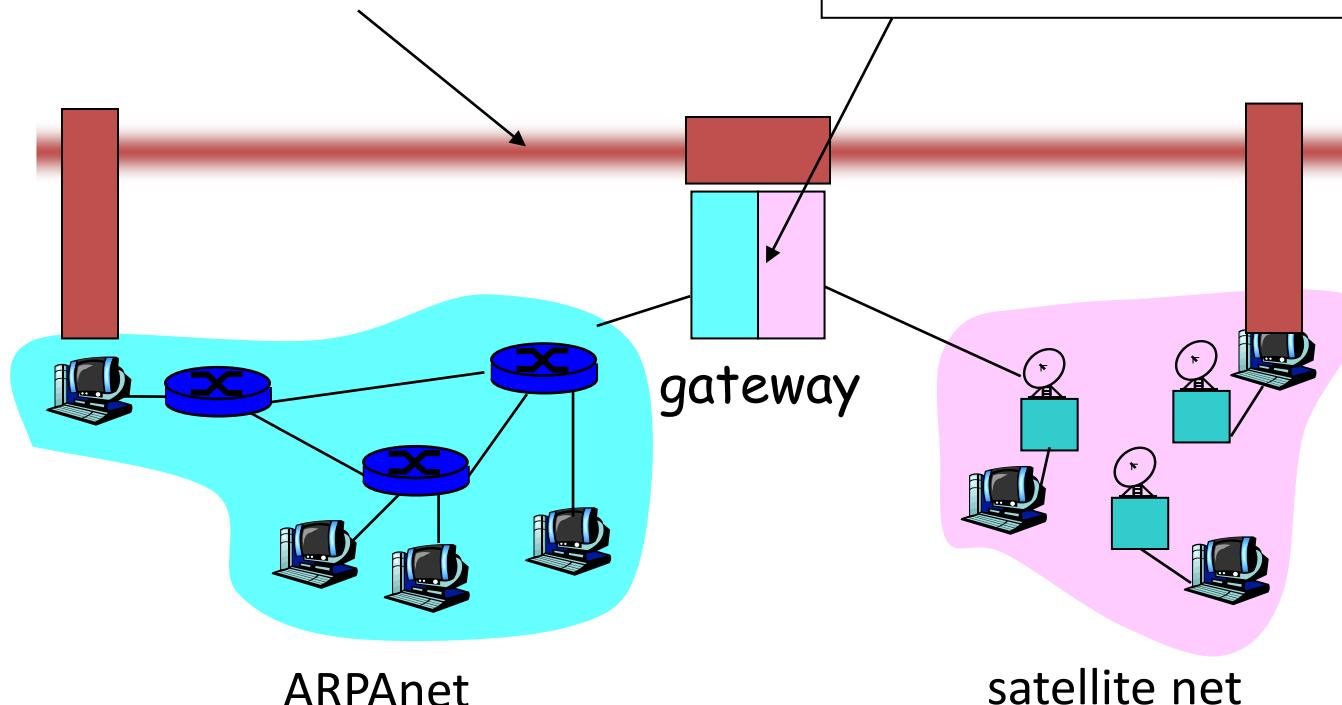
# Recall (again ☺) the Internet: bridging/virtualizing networks

Internet layer :

- internetwork connects seamlessly underlying heterogeneous local networks
- Create network of networks; in the spirit of simplicity and collaboration

Gateway:

- “embed internetwork packets in local packet format”
- route (at inter-network level) to next gateway



1974



TCP / IP defined by Vint Cerf & Bob Kahn



2004: both received the Turing Award

(proposed in 70's, standard 80's)  
"A Protocol for Packet Network Intercommunication", V. Cerf, R. Kahn, IEEE Transactions on Communications, May, 1974, pp. 637-648.

# Evolotion of Internet layers&protocols

**Application:** protocols supporting *network applications*

http (*web*), smtp (*email*),

**virtualization&p2p, streaming, CDN, http2/3/QUIC...**

**transport:** process2process (end2end) data transfer

UDP, TCP,

**QUIC**

**network:** routing of datagrams (independent data-packets), connecting

different physical networks

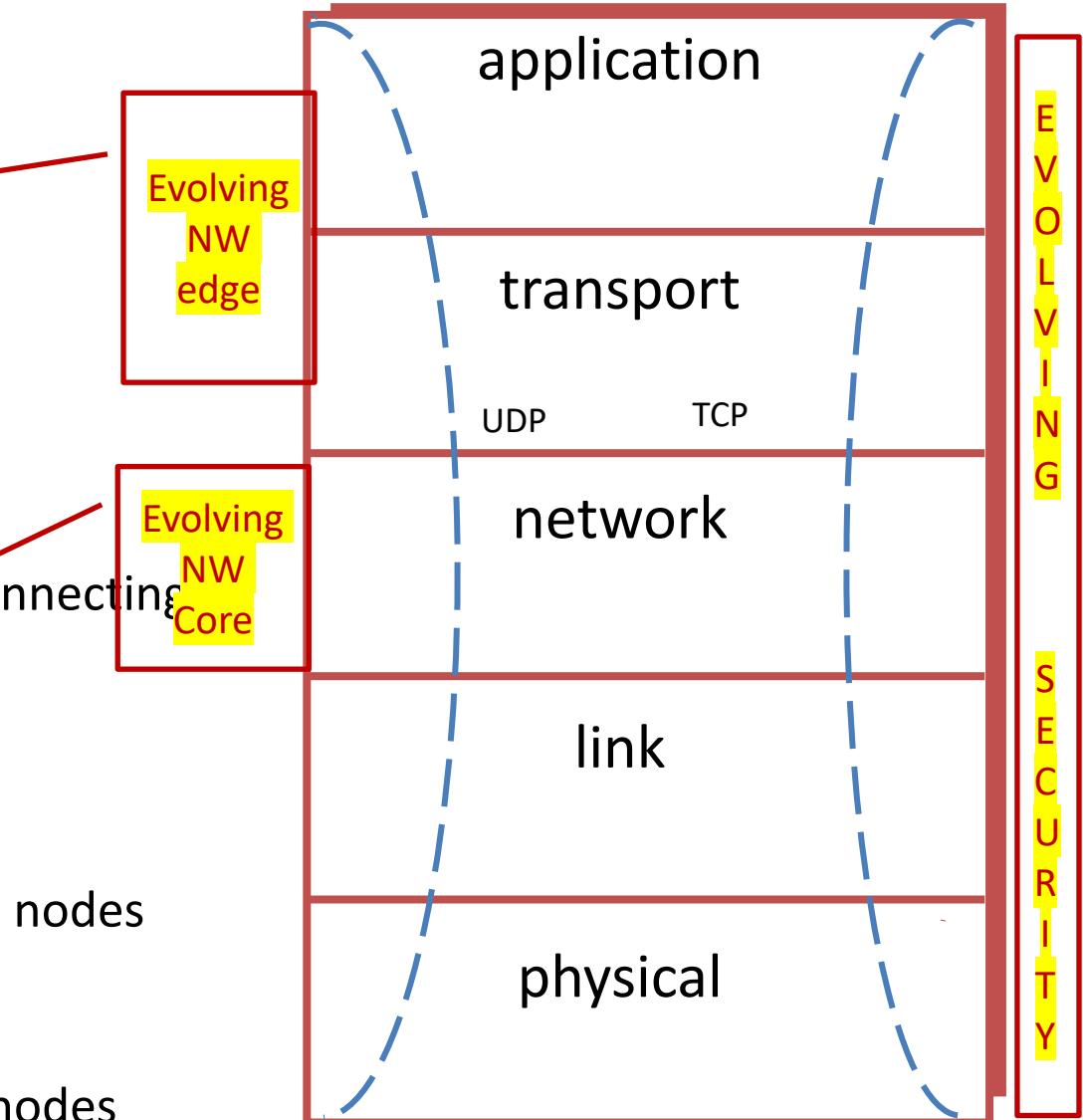
**IP addressing, routing protocols,**

**virtualization, virtualization, ...**

**link:** data transfer between neighboring (physically connected) nodes

Ethernet, WiFi, ...

**physical:** bit-transmission on physical medium -- neighboring nodes



# Roadmap – Internet evolution @edge



## P2P apps and overlays for info sharing Ch: 2.5

### Media Streaming Ch 2.6 , 3.8

- Application classes, challenges
- Representative technology
  - Recovery from jitter
  - Streaming protocols and new proposals

CDN: overlays infrastructure for content delivery

*around year 2000: NW bandwidth & Internet rapid growth...*

# On distributed, P2P applications

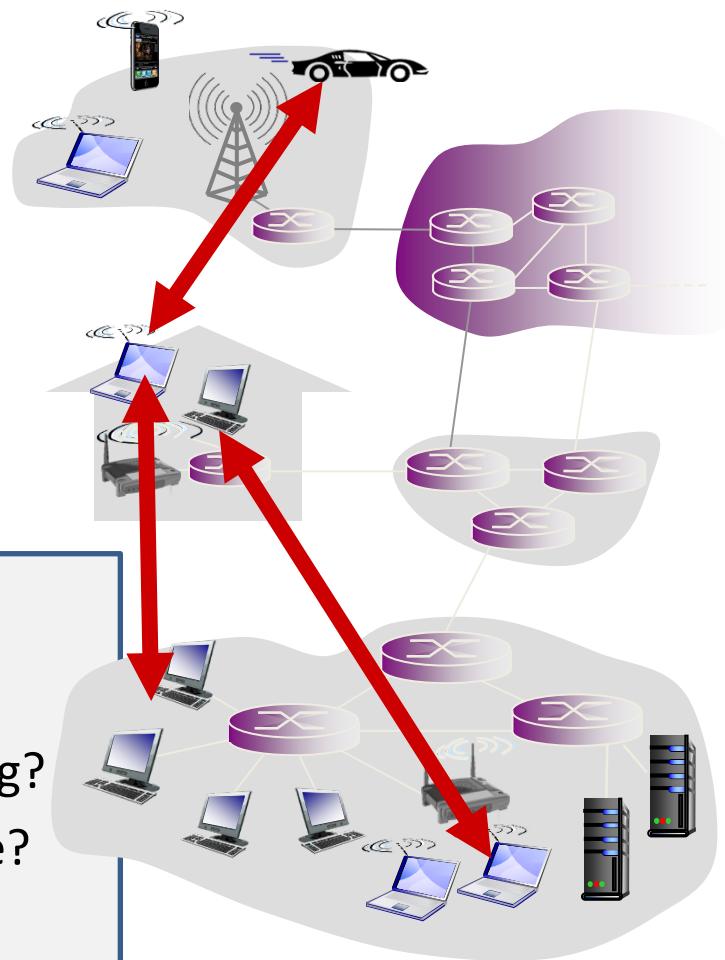
- no always-on server
- peers request&provide service from/to other peers
  - self scalability – new peers bring new service capacity, and new service demands
- peers are intermittently connected and may change IP addresses (complex management)

*examples:*

- **file distribution/sharing**
- Streaming multimedia

## Common Primitives in file-sharing p2p apps:

- **Join:** how to begin participating?
- **Publish:** how to advertise a file?
- **Search:** how to find a file?
- **Fetch:** how to retrieve a file?



# P2P: once upon a time ...centralized directory

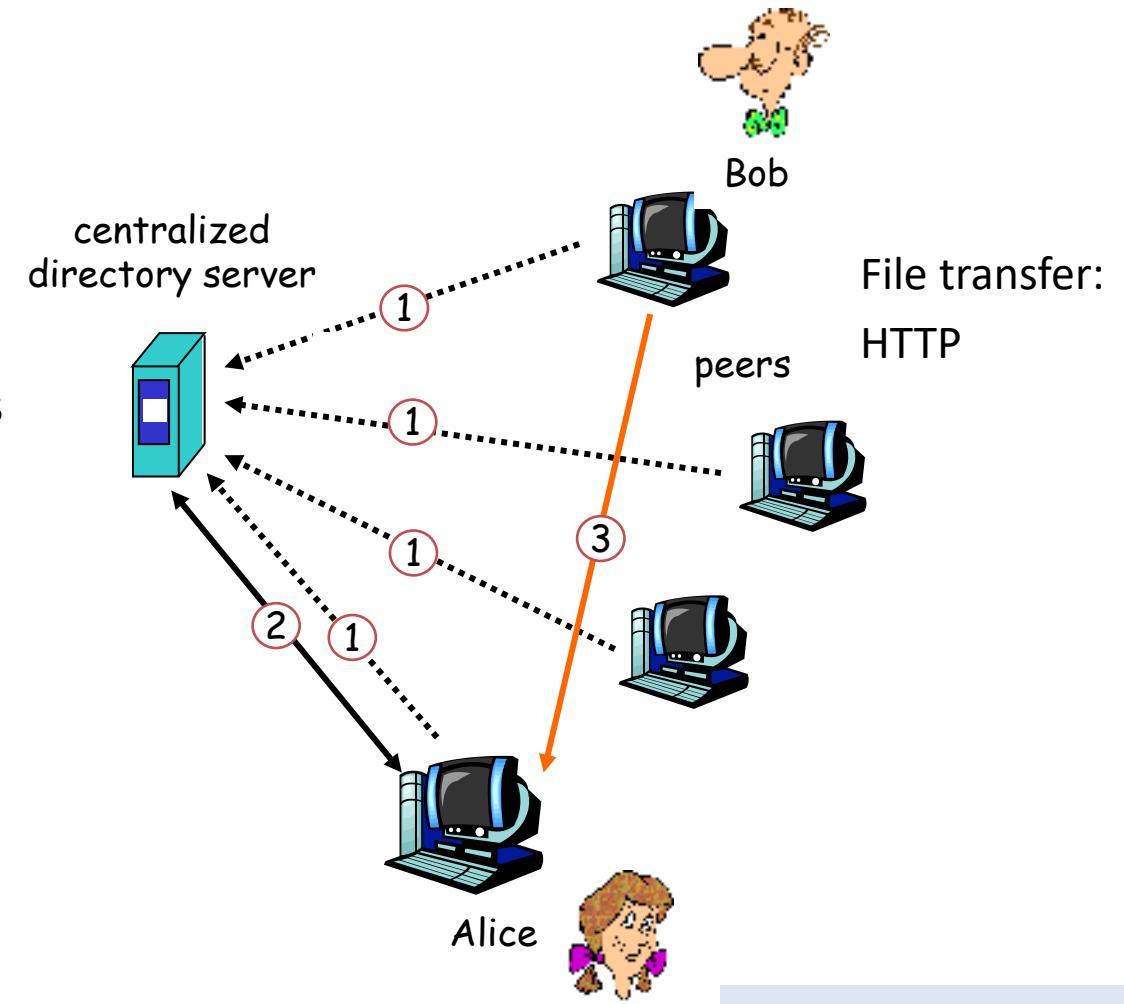
original “Napster” design (1999, Shawn Fanning & Saun Parker):

1) when peer connects, it informs central server:

- IP address, content

2) Alice queries directory server for “LetItBe”, get’s Bob’s IP address

3) Alice requests file from Bob



Q: What is p2p in this?

# Roadmap -- Internet evolution @edge



P2P apps and overlays for info sharing

- **Collaborate/form-overlays to find content:**
  - Collaborate/form-overlays to fetch content

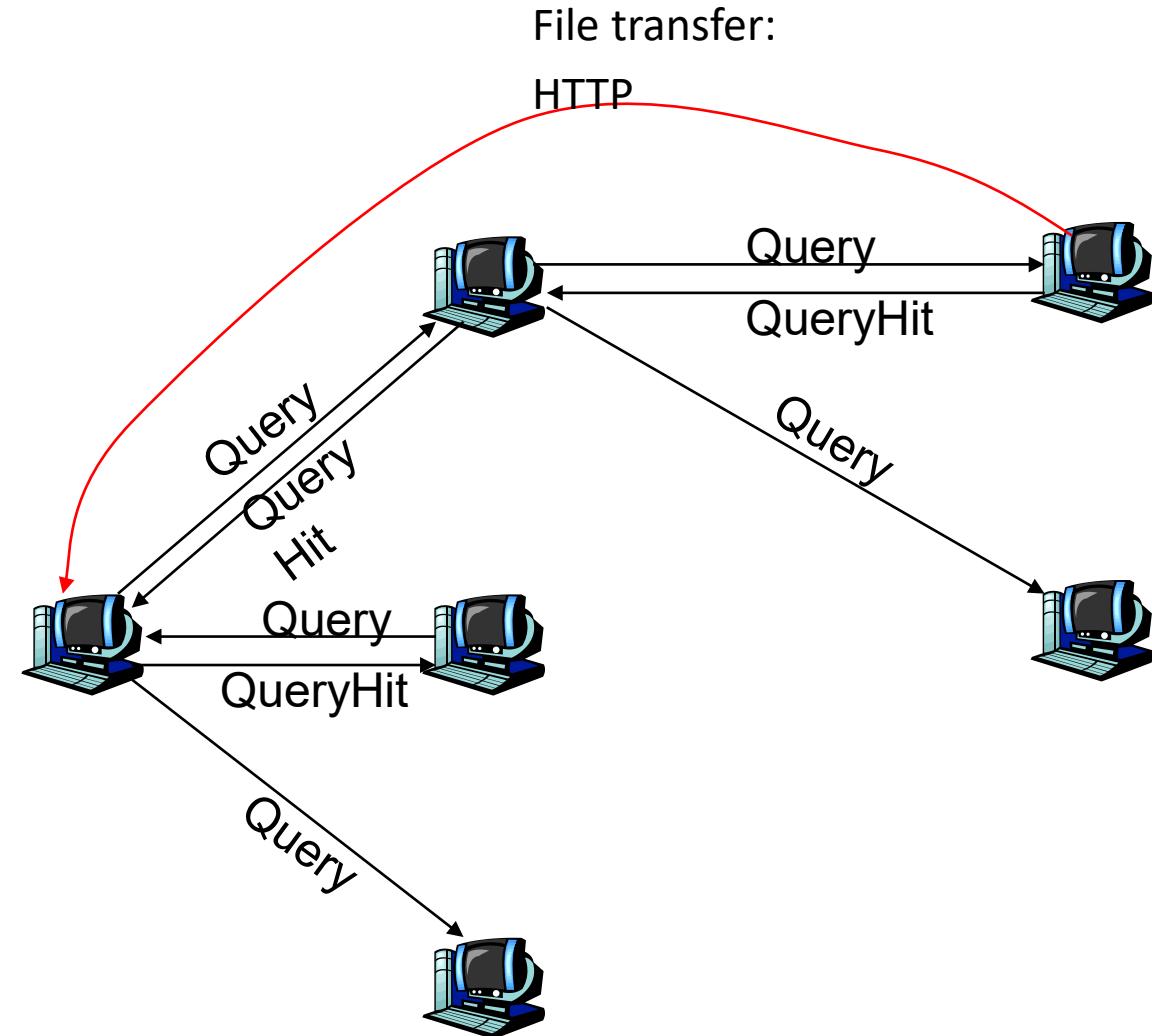
## Media Streaming

- Application classes, challenges
- Representative technology
  - Recovery from jitter
  - Streaming protocols and new proposals

CDN: overlays infrastructure for content delivery

# P2P Gnutella (no directory): protocol

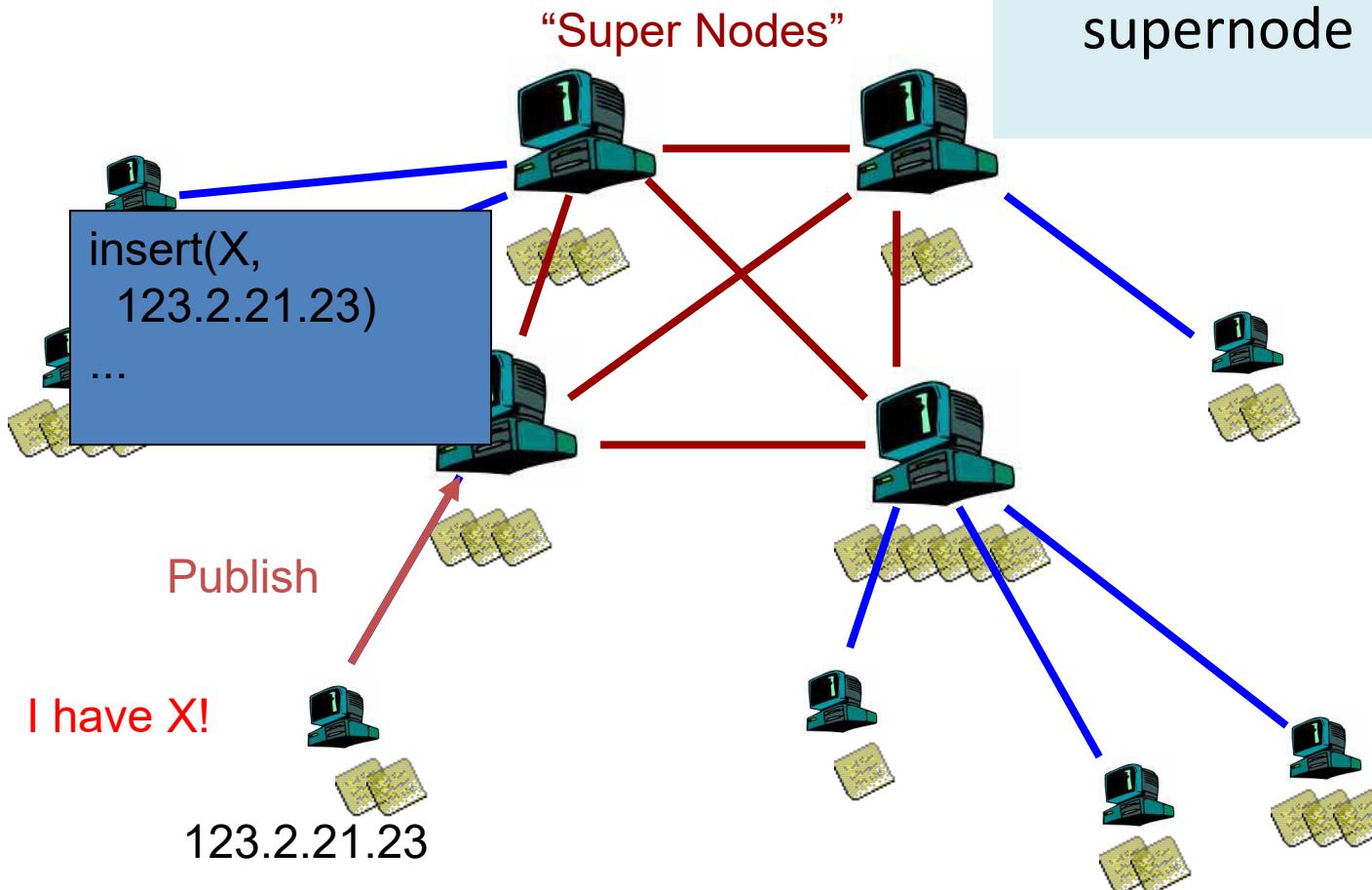
- **Join:** on startup, client connects to a few other nodes (learn from **bootstrap-node**); these become its “neighbors” (**overlay!! ☺**)
- **Publish:** no need
- **Search:** Query Flooding:
  - ask “**neighbors**”, who ask their neighbors, and so on; when/if found, reply to sender.
- **Fetch:** get the file directly from peer



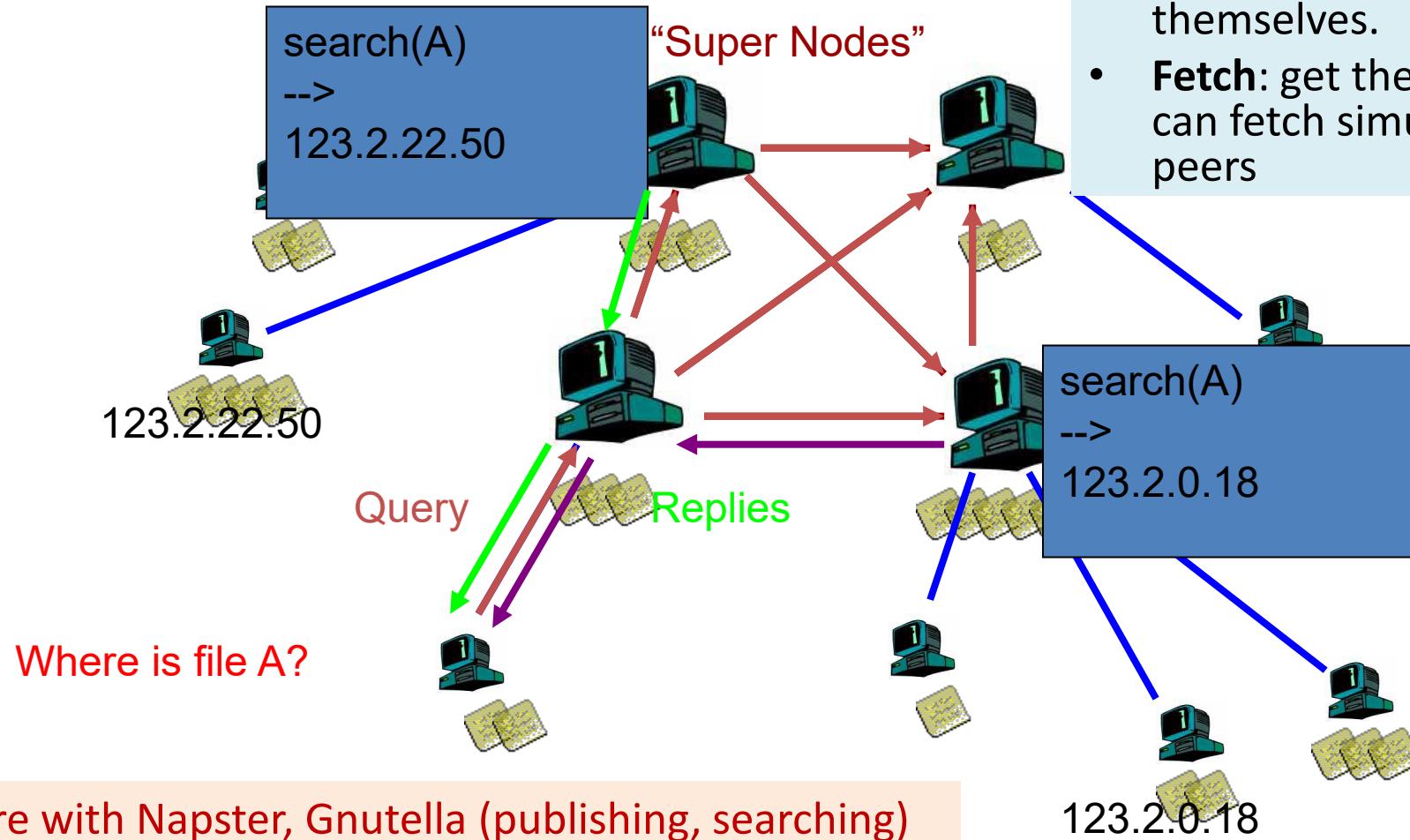
**Q:** pros & cons of napster & gnutella?

# P2P KaZaA : distributed directory & “Smart” Query Flooding

- **Join:** client contacts a “supernode”
- **Publish:** send list of files to supernode



# P2P KaZaA : distributed directory & “Smart” Query Flooding



# Another Distributed Approach (DHT)

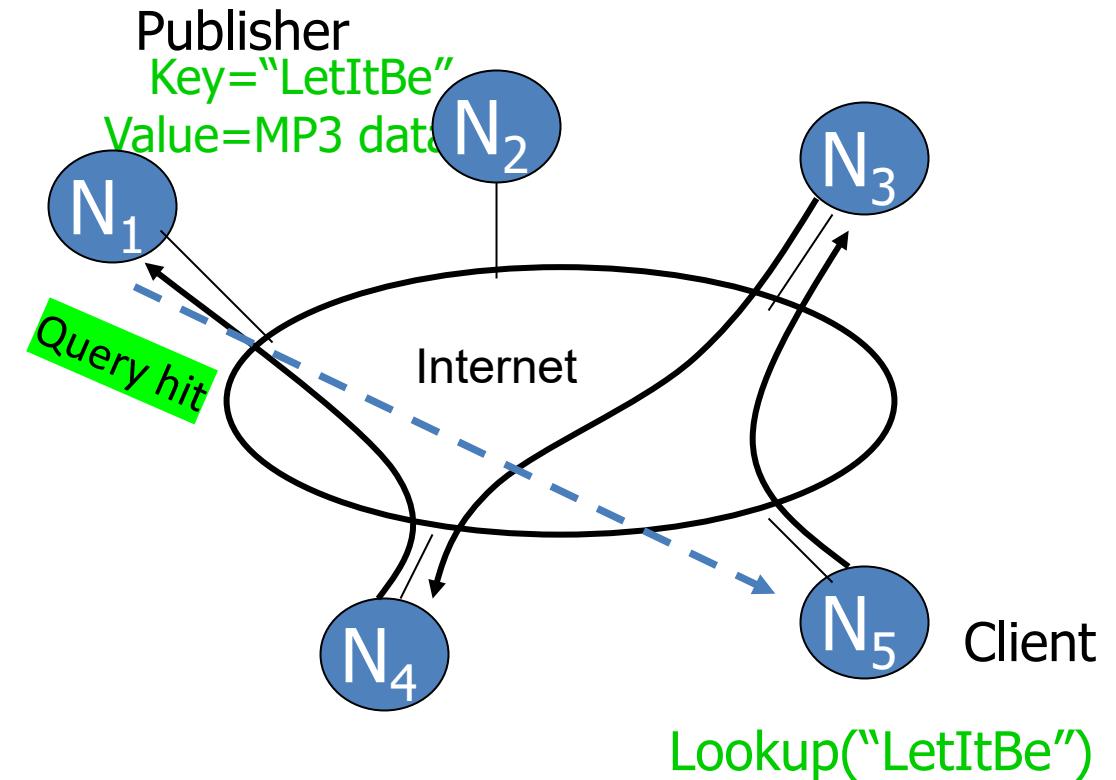
[Noticed the similarity with data-bases?]

IDEA: maintain a lookup data structure (distributed "hash-table" DHT, i.e. a mapping data-structure

Each node (peer) is responsible for

- Maintaining **subset of indexes to the content** (for those keys that are (hash)mapped to the node)
- Maintaining&knowing its **overlay neighbours**, to **forward the question** for any possible key

The **overlay** (to forward queries) can have a topology of a **ring** (eg Chord, with shortcuts for binary search) or **cube, tree, etc**



# Roadmap -- Internet evolution @edge



P2P apps and overlays for info sharing

- **Collaborate/form-overlays to find content:**
- **Collaborate/form-overlays to fetch content**

## Media Streaming

- Application classes, challenges
- Representative technology
  - Recovery from jitter
  - Streaming protocols and new proposals

CDN: overlays infrastructure for content delivery

# Second generation p2p: focus on fetching

## Motivation:

- Popularity exhibits temporal locality (Flash Crowds)
- Can bring file “provider” to “its knees”



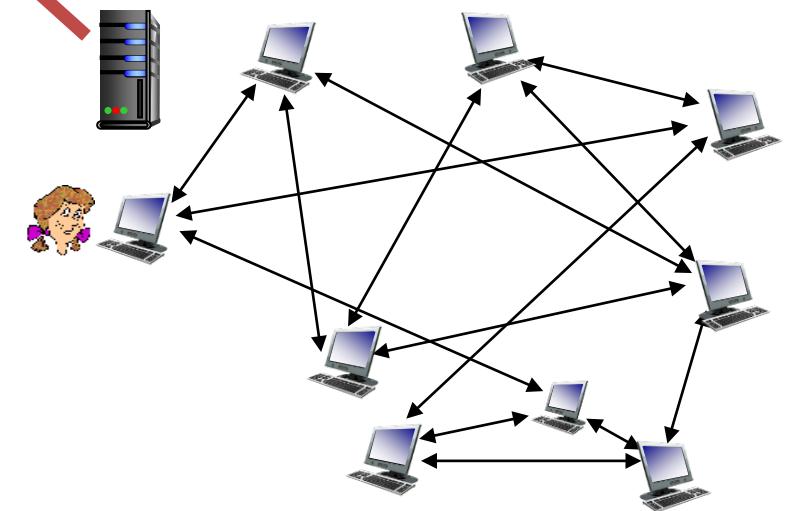
**torrent:** group of peers exchanging chunks of a file

## Idea:

- Files “chopped” in chunks; fetch from many sources (**swarming**)
- **Overlay:** nodes “hold hands” (i.e. maintain a connection) with those who share chunks

- **Join:** contact a server, aka “**tracker**” get a list of peers.
- **Publish:** can run a tracker server.
- **Search:** Out-of-band. E.g., use search-engine, or DHT, ... to **find a tracker**, which **gives list of peers to contact**
- **Fetch:** Download chunks from several of peers. Upload chunks you have to them

**tracker:** tracks peers participating in torrent



Architecture used by publishers to distribute software, other large files

# Roadmap



P2P apps and overlays for info sharing

- Collaborate/form-overlays to *find* content:
- Collaborate/form-overlays to *fetch* content

## Media Streaming

- Application classes, challenges
- Representative technology
  - Recovery from jitter
  - Streaming protocols and new proposals

CDN: overlays infrastructure for content delivery

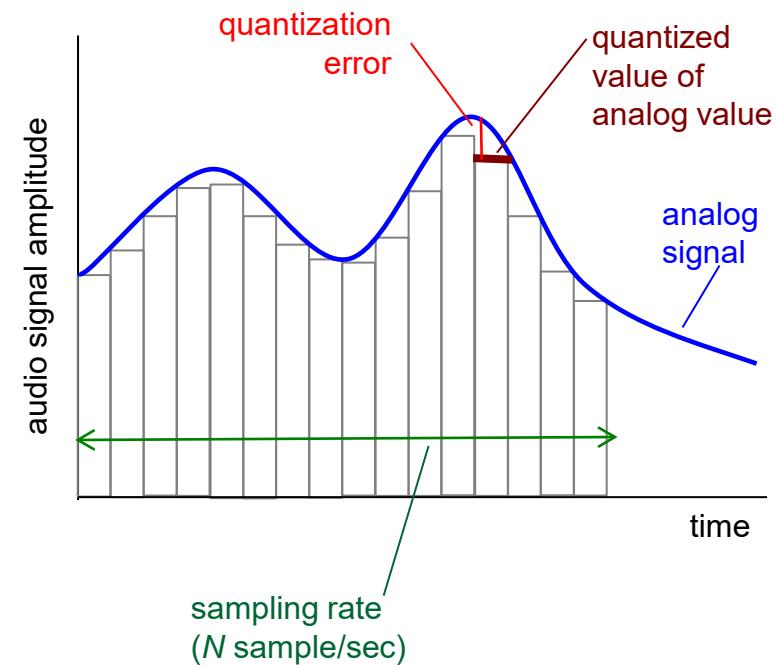
*around year 2000: NW bandwidth & Internet rapid growth...*

# Multimedia: audio

- analog audio signal sampled at constant rate
  - telephone: 8,000 samples/sec
    - 256 quantized levels => 64,000 bps
  - receiver converts bits back to analog signal:

## example rates

- CD: 1.411 Mbps
- MP3: 96, 128, 160 kbps
- Internet telephony: 5.3 kbps and up



# Multimedia: video

- video: sequence of images (arrays of pixels) displayed at constant rate
  - e.g. 24 images/sec

*CBR: (constant bit rate):* encoding rate fixed

*VBR: (variable bit rate):* encoding rate changes as amount of spatial, temporal coding changes

*examples:*

MPEG<sup>(\*)</sup> 1 (CD-ROM) 1.5 Mbps

MPEG2 (DVD) 3-6 Mbps

MPEG4 (often used in Internet, 64Kbps-10's Mbps)

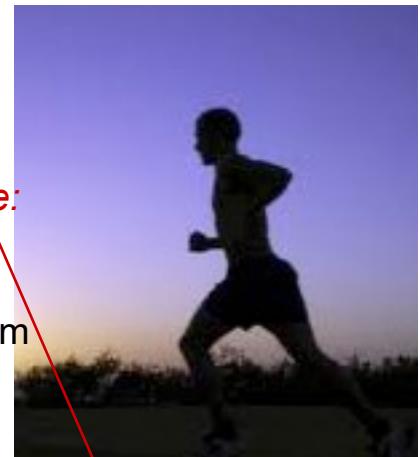
<sup>(\*)</sup> ISO/IEC Moving Picture Experts Group (MPEG)

*spatial coding example:* instead of sending  $N$  values of same color (all purple), send only two values: color value (*purple*) and number of repeated values ( $N$ )



frame  $i$

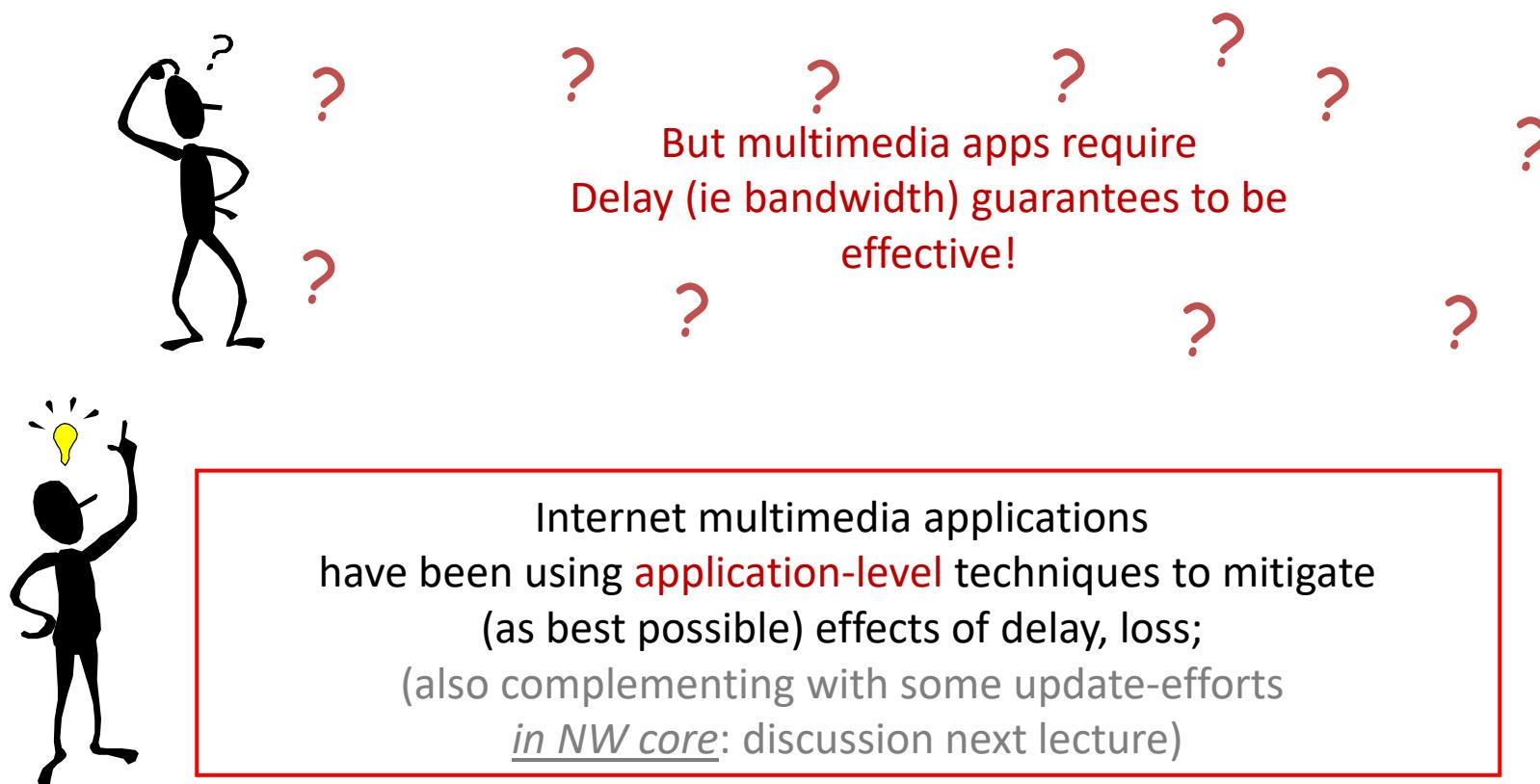
*temporal coding example:* instead of sending complete frame at  $i+1$ , send only differences from frame  $i$



frame  $i+1$

# Recall Internet's main transport services: TCP, UDP

*no* guarantees on delay....



# Solution approaches in Internet for serving streaming traffic

---

- Several applications used **UDP** to avoid TCP's ack-based progress (and slow start), but...
- Buffer content at client and control playback to **remedy jitter**
- Different **error control** methods (**not** ack-based)
- Exhaust uses of **caching, proxys**, etc
- Adapt **compression** level to available bandwidth
- add **more bandwidth**

+ after years with “bags of tricks” at applications, recently: **protocol evolution at the transport+application layers**

+ some update efforts in the NW core; subsequent lecture

# Roadmap -- Internet evolution @edge



## P2P apps and overlays for info sharing

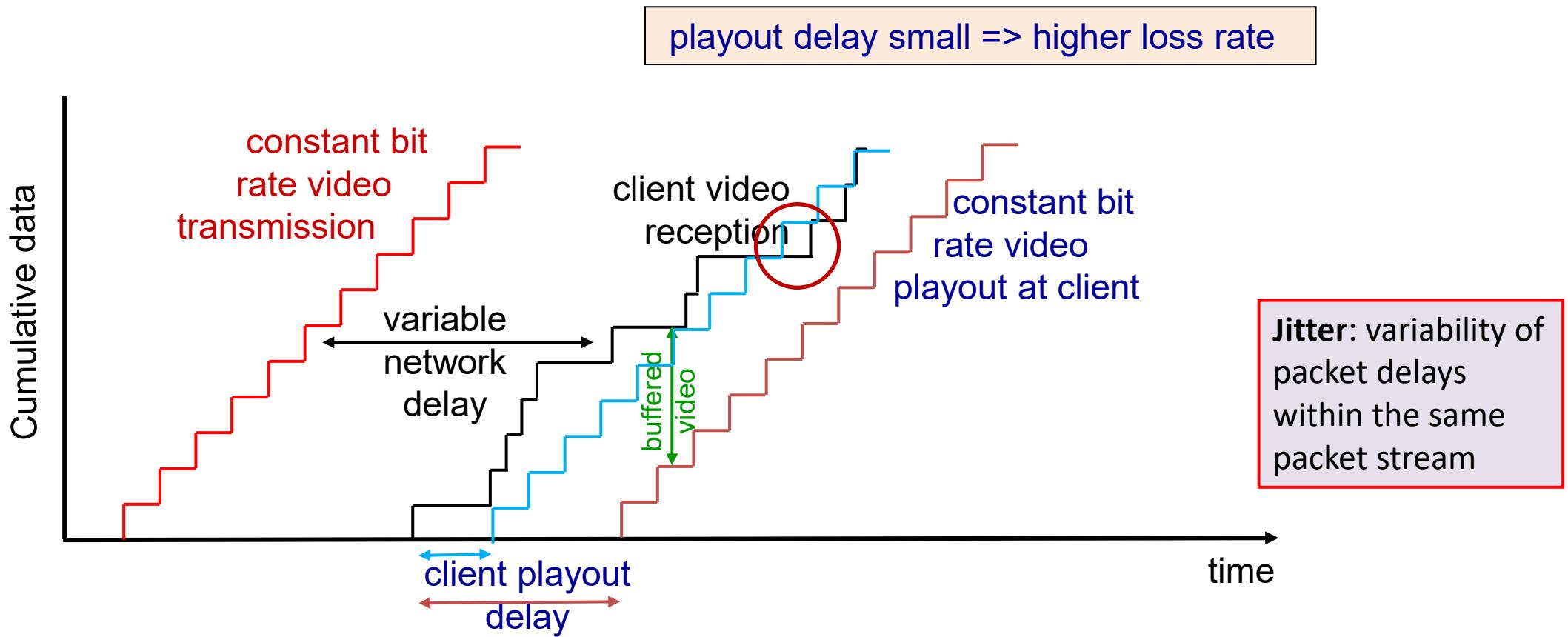
- Collaborate/form-overlays to *find* content:
- Collaborate/form-overlays to *fetch* content

## Media Streaming

- Application classes, challenges
- Representative technology
  - Recovery **from jitter**
  - Streaming protocols and new proposals

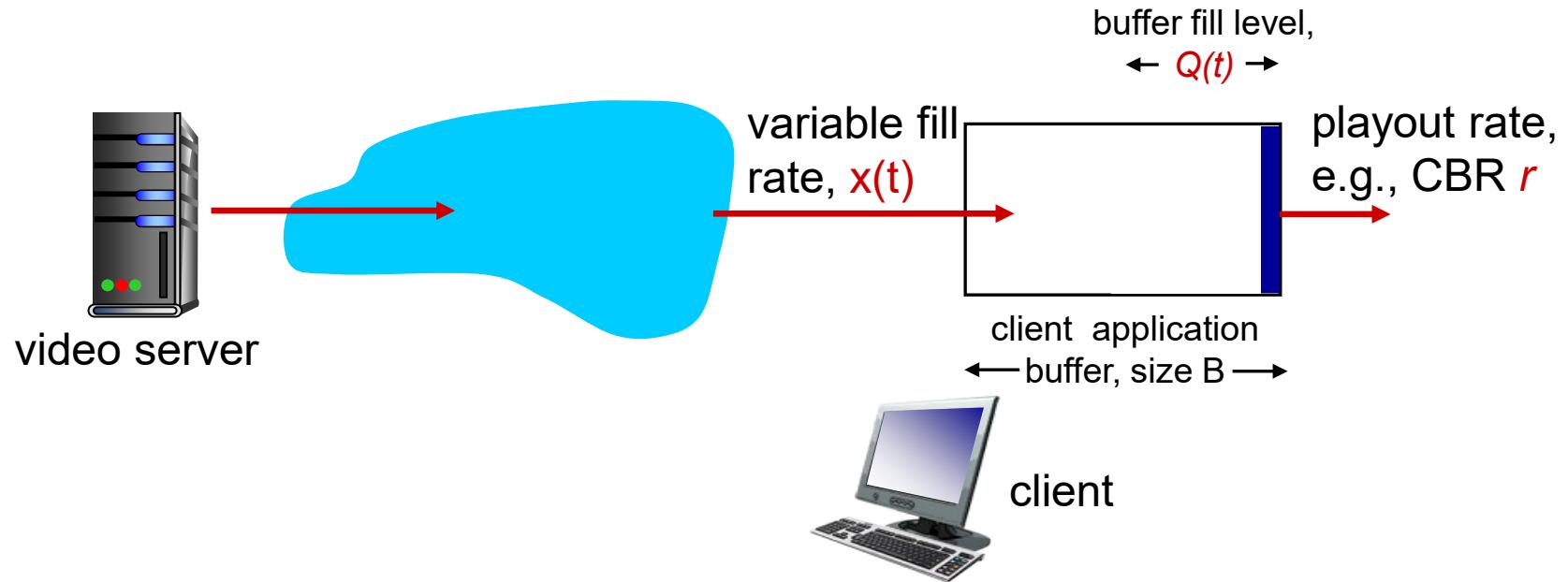
CDN: overlays infrastructure for content delivery

# Streaming: recovery from jitter



*client-side buffering and playout delay:* compensate for network-added delay & jitter

# Client-side buffering, playout



1. Initial fill of buffer until ...
2. ... playout begins at  $t_p$ ,
3. buffer fill level varies as fill rate  $x(t)$  varies, but playout rate  $r$  is constant

# Roadmap -- Internet evolution @edge



## P2P apps and overlays for info sharing

- Collaborate/form-overlays to *find* content:
- Collaborate/form-overlays to *fetch* content

## Media Streaming

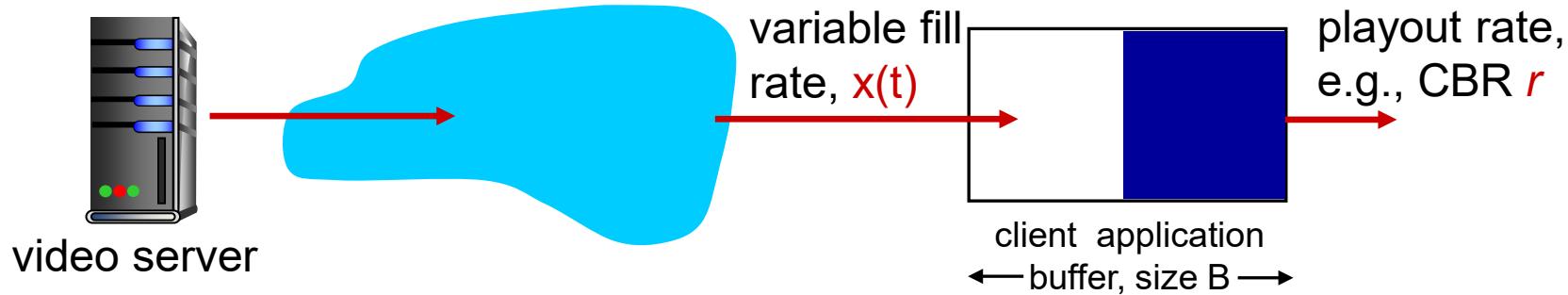
- Application classes, challenges
- Representative technology
  - Recovery from jitter
  - Streaming protocols and new proposals (App -- Transport Layer collaboration)
  - CDN: overlays infrastructure for content delivery

# Streaming multimedia: UDP?

---

- server sends at rate appropriate for client
  - often: send rate = encoding rate
  - send rate can be oblivious to congestion levels (**is this good? selfish?**)
- short playout delay to remove network jitter
- **BUT also:** UDP may not go through firewalls

# Client-side buffering, playout using UDP



*playout buffering: average fill rate ( $\bar{x}$ ), playout rate ( $r$ ):*

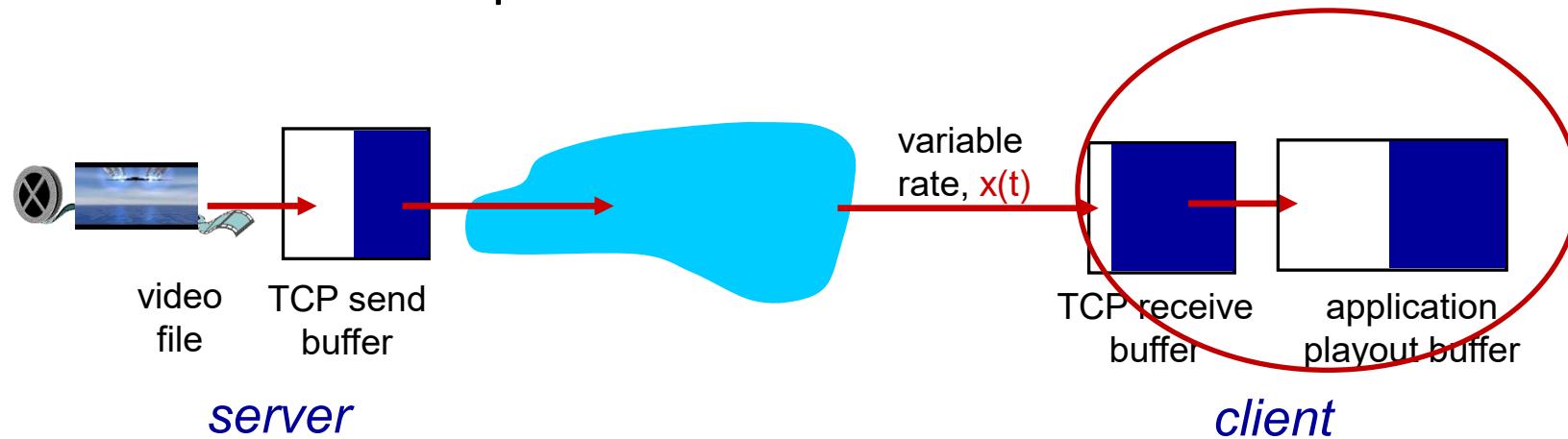
- $\bar{x} < r$ : buffer eventually empties (causing freezing of video playout until buffer again fills)
- $\bar{x} > r$ : need to have enough buffer-space to absorb variability in  $x(t)$

*playout delay tradeoff:*

- *small delay*: buffer can get easier empty (packets can be “lost”)
- larger delay: longer time until user begins watching, buffer can get full (thus also “lose pkts”)

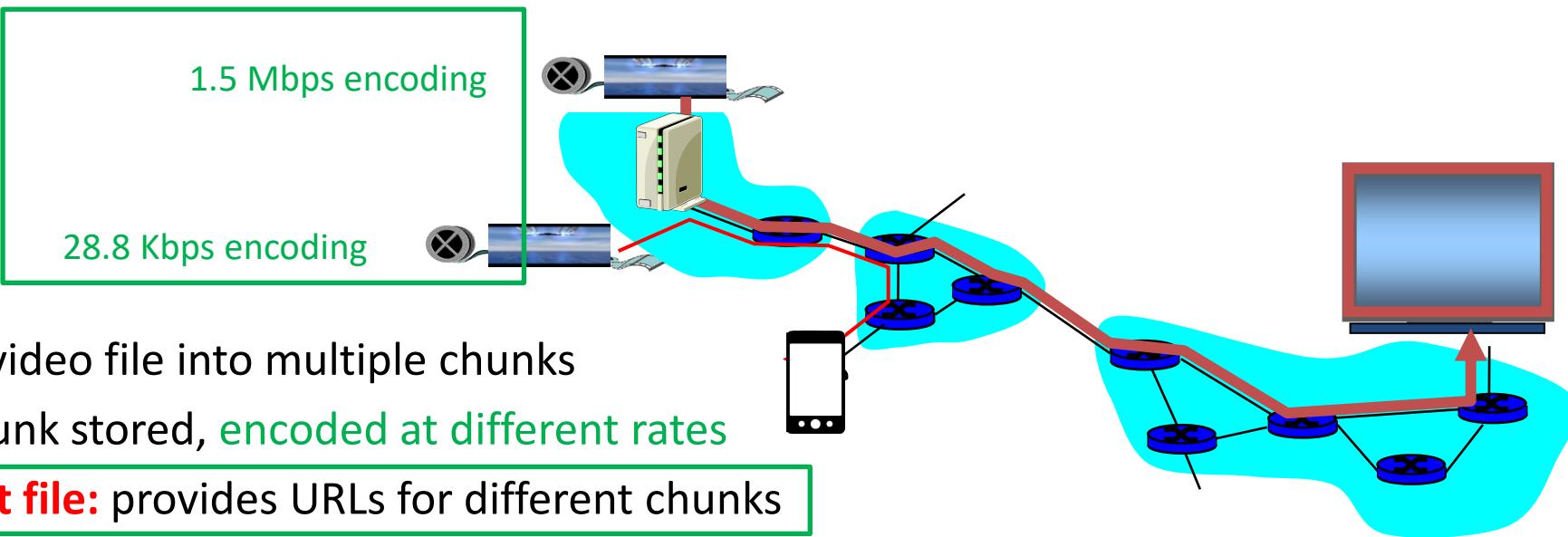
# Streaming multimedia through TCP

- multimedia file retrieved via HTTP GET
  - HTTP/TCP collaborates better with firewalls
- send at maximum possible rate under TCP



- fill rate fluctuates due to TCP congestion control, retransmissions, in-order delivery
- **larger/adaptive playout delay:** to smooth TCP saw-tooth delivery rate
- But ... not enough to deal with variable bandwidth...

# Streaming multimedia: DASH: Dynamic, Adaptive Streaming over HTTP



*server:*

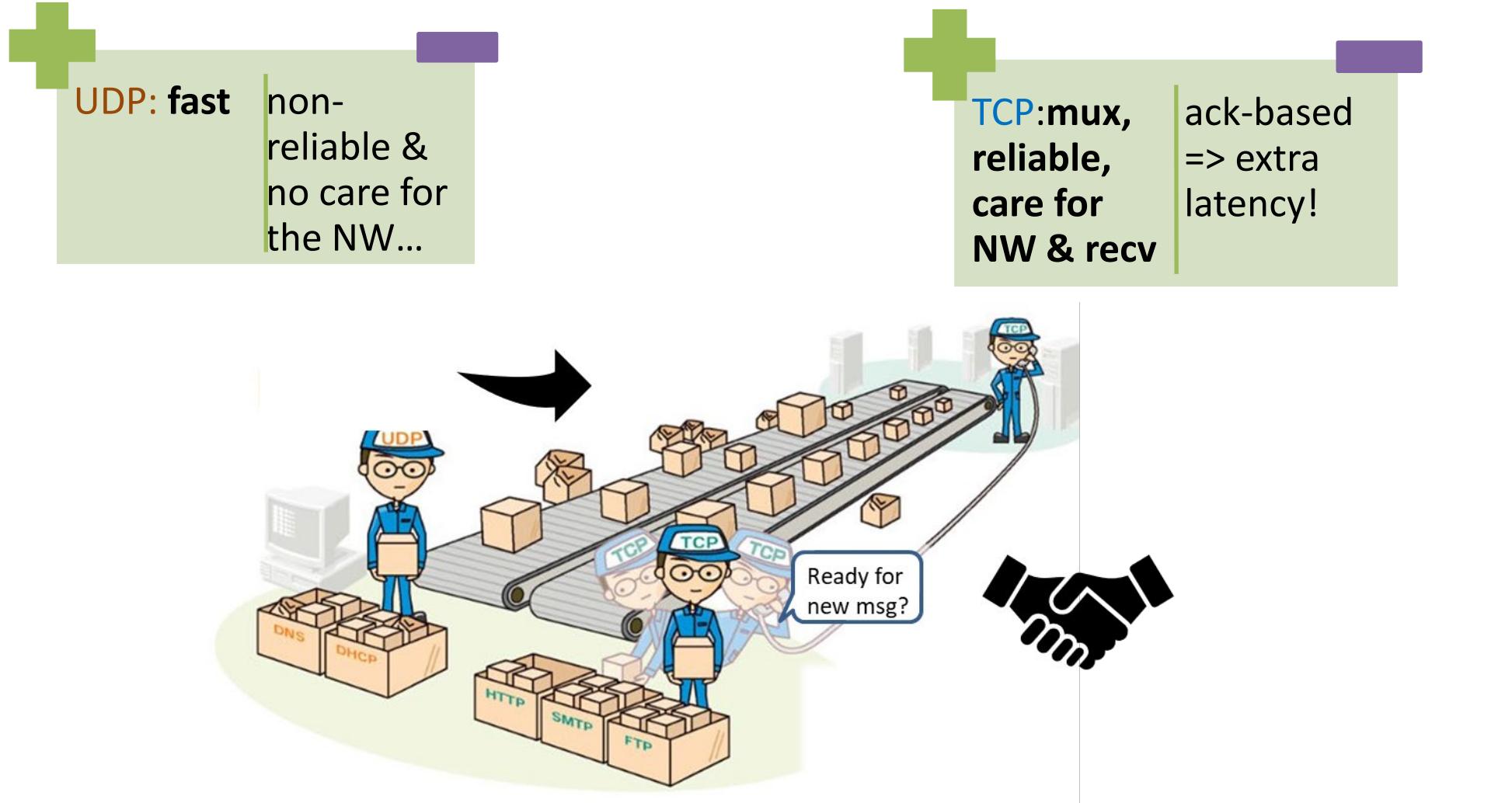
- divides video file into multiple chunks
- each chunk stored, **encoded at different rates**
- **manifest file:** provides URLs for different chunks

*"intelligence" @client:*

- periodically **measures server-to-client bandwidth**
- consulting manifest file, **requests one chunk at a time, at appropriate coding rate and appropriate time**
  - can choose **different coding rates at different points** in time (depending on available bandwidth)
  - decides **when to request chunk** (to avoid buffer starvation and overflow)

**Streaming video = encoding + DASH + playout buffering**

# More on recent evolution ...



**Challenge:** can we build sufficient reliability and care for NW but FAST (eg not Ack-based, over UDP)?  
i.e. can we have TCP-like behaviour with less of the "bureaucracy" of TCP?

# QUIC (Quick UDP Internet Connections)

Announced [Google 2013] to improve performance of apps that used TCP

- does bandwidth estimation to avoid congestion (**congestion avoidance @application space**)
- supports **MUXed connections over UDP**
- + designed for **security** protection equivalent to SSL

- 2015: *Internet Draft* of a specification for QUIC submitted to IETF for standardization
- *QUIC working group*: multipath support & forward error correction (FEC)
- 2018: HTTP & QUIC Working Group made an official request for the (“combined”) protocol HTTP/3 before standardization
- 2021: IETF standardized the protocol (RFC 8999-9002)

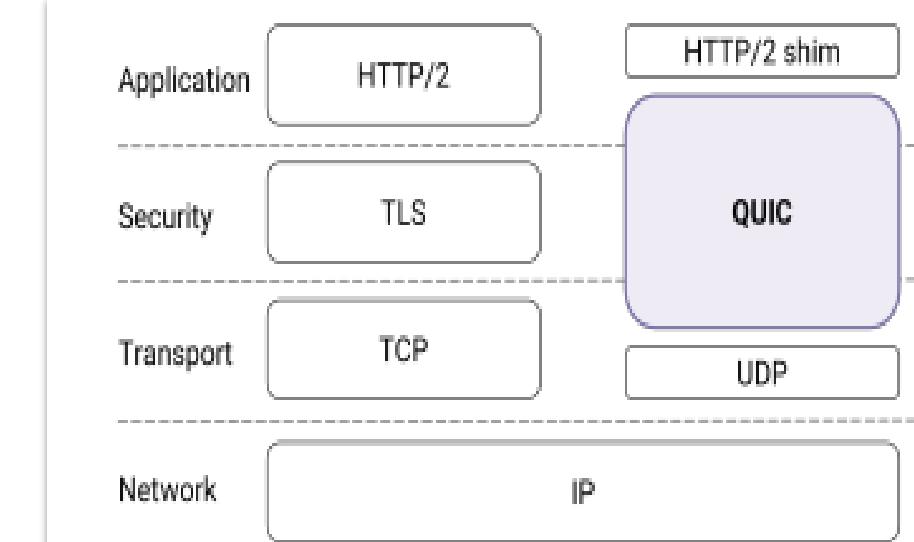


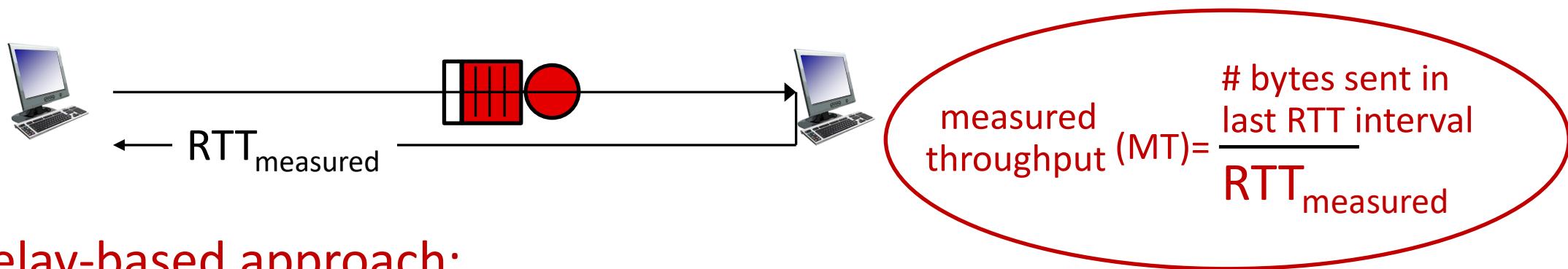
Figure 1: QUIC in the traditional HTTPS stack.

Quic; ACM SIGCOMM '17 <https://doi.org/10.1145/3098822.3098842>

Idea known for a while: see works by Sally Floyd et.al, on TCP-friendliness (“Her work on congestion control helped keep the internet working for everyone.” [CaroleLeita]); e.g.: DCCP congestion control without reliability :. SIGCOMM Comp. Commun. 2006 <http://doi.acm.org/10.1145/1151659.1159918>

# Update: Delay-based congestion control

Keep sender-to-receiver pipe “just full enough, but no fuller”: keep bottleneck link busy, but avoid high delays/buffering



Delay-based approach:

■ ~~RTT<sub>min</sub> - minimum observed RTT (uncongested path)~~

■ ~~Estimated Uncongested Throughput (UT) = cwnd/RTT<sub>min</sub>~~

if MT ~ UT : increase cwnd linearly

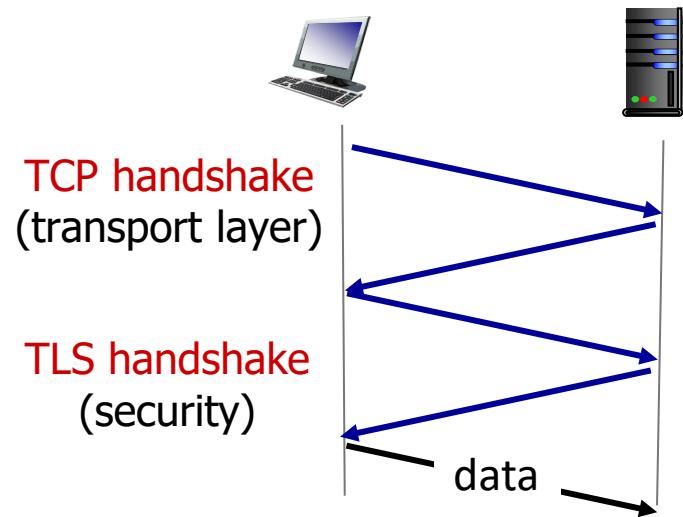
if MT << UT decrease cwnd linearly

/\* path not congested \*/

/\* path is congested \*/

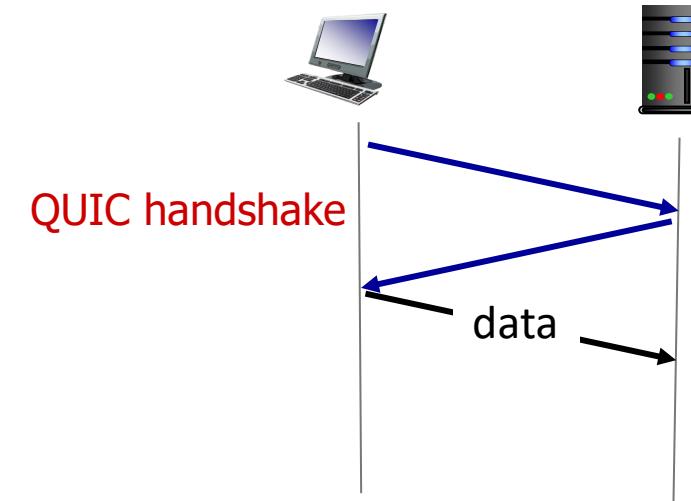
BBR (Bottleneck Bandwidth and Round-trip propagation time):  
deployed on Google's (internal)  
backbone network  
Cf: Cardwell, Neal, et al. "BBR  
congestion-based congestion  
control." Communications  
ACM 2017

# QUIC: Connection establishment



TCP (reliability, congestion control state) + TLS (authentication, crypto state)

- 2 serial handshakes



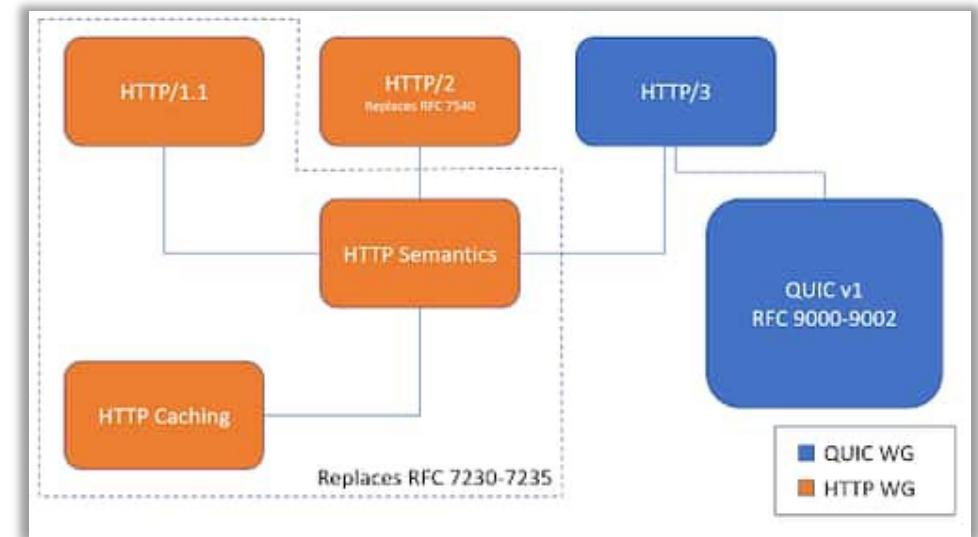
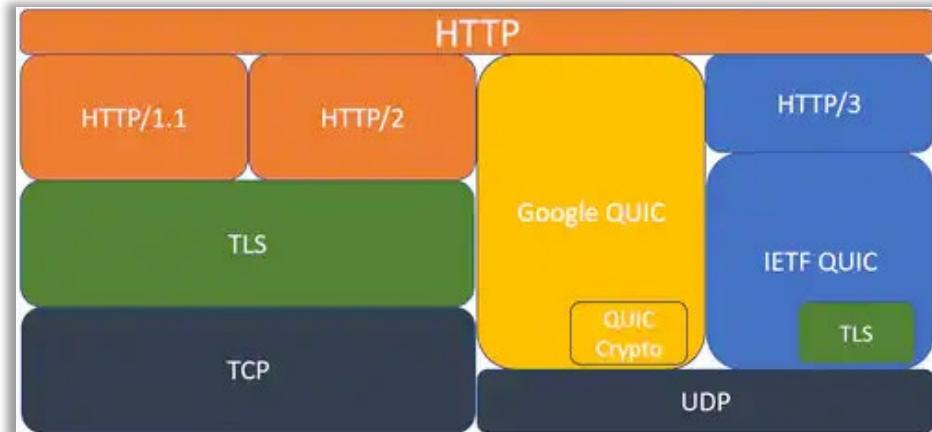
QUIC: reliability, congestion control, authentication, crypto state

- 1 handshake

# HTTP/2 & HTTP/3 over QUIC

- Derived from SPDY (in Google's proprietary NWs; other proprietary NWs were also doing similarly)
- New: how data is framed and transported, e.g.:
  - Compression & “**minification**” of resources e.g. images, scripts (as in DASH)
  - **server can also "push"** content, i.e. respond with more data than the client requested
  - **prioritization** of requests, **multiplexing** (see next slides -->)
- 2022: IETF published it as a Proposed Standard (RFC 9114)

overwhelming complexity [P.H. Kamp, ACM queue 2015]

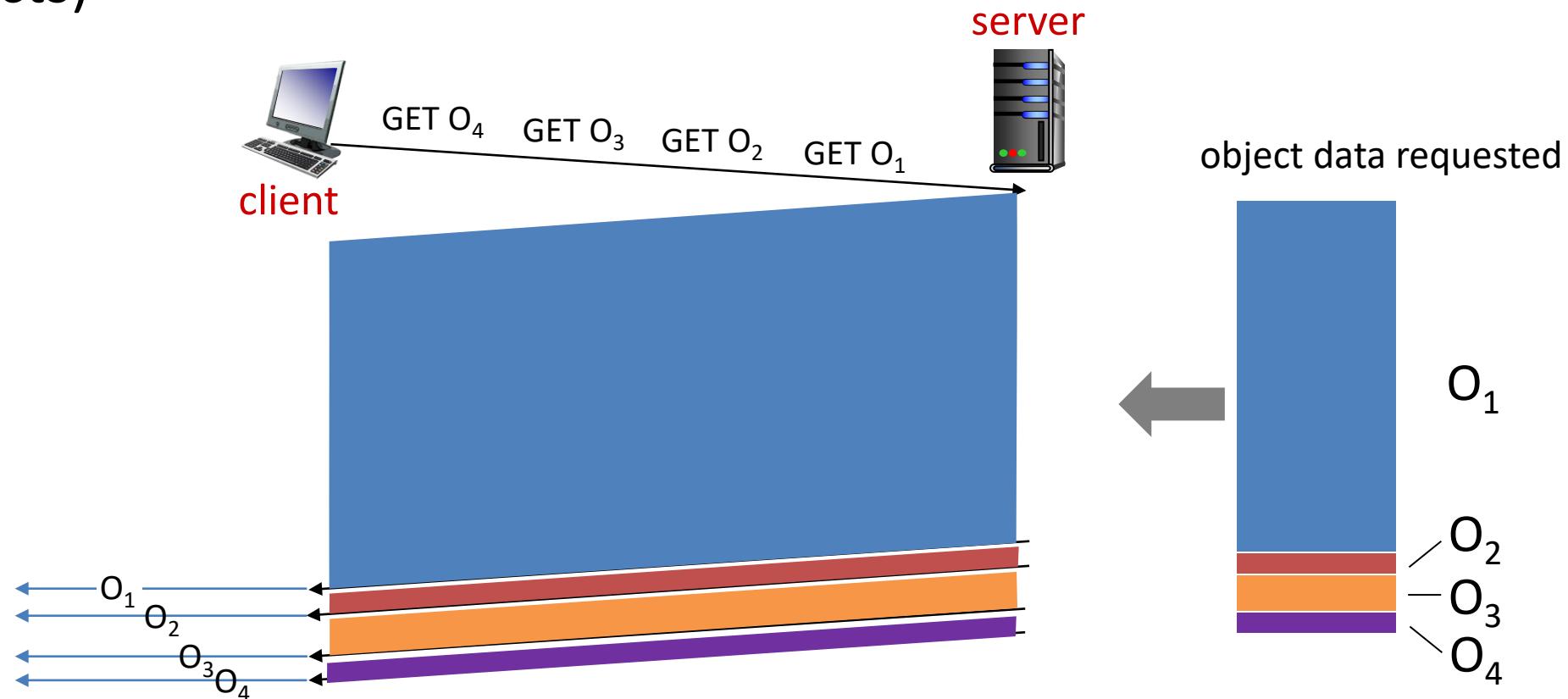


Mike Bishop, June 2021

<https://www.akamai.com/blog/performance/http3-and-quic-past-present-and-future>

# Recall: Problem: HOL (head-of-line) blocking

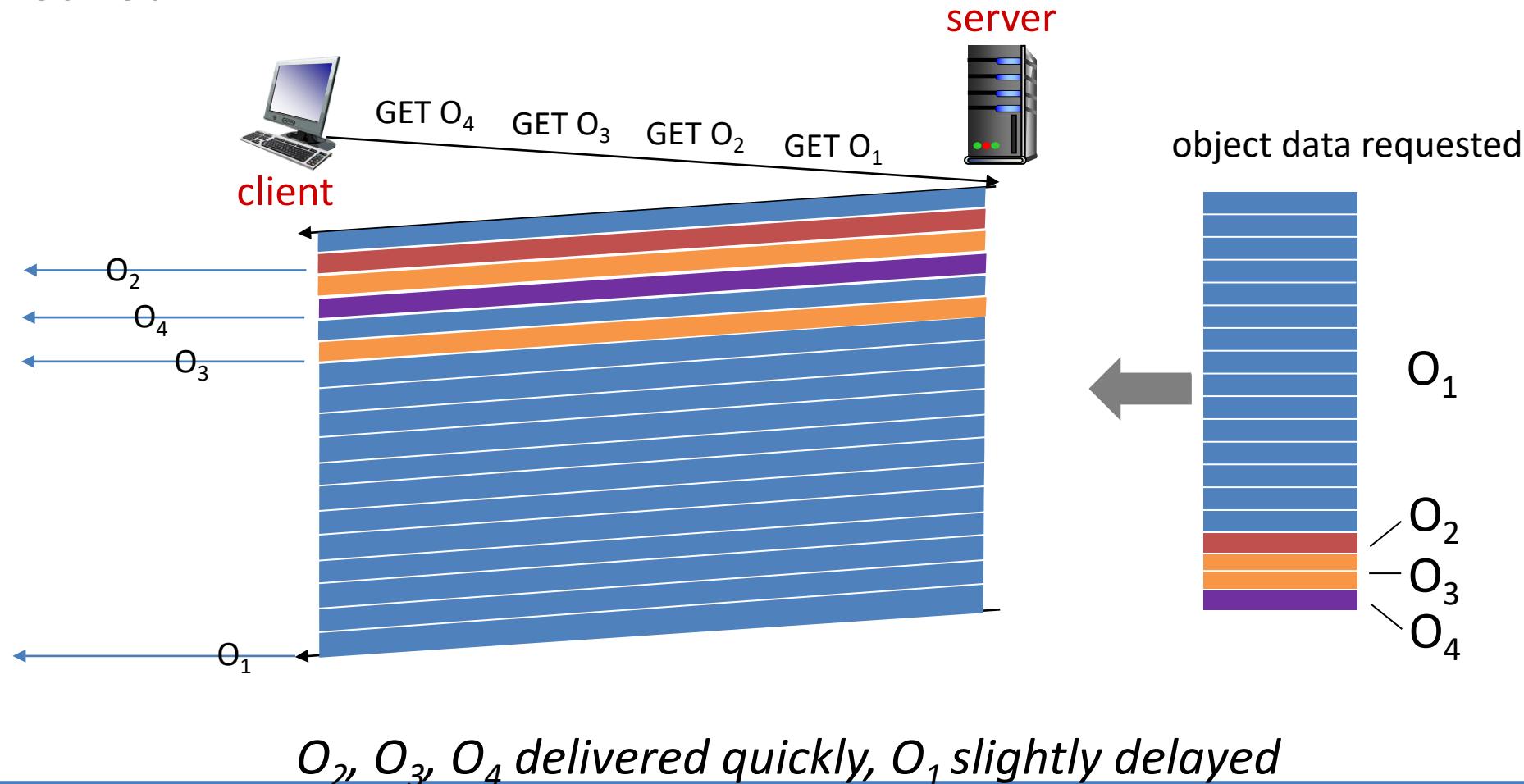
HTTP 1.1: client requests 1 large object (e.g., video file, and 3 smaller objects)



*objects delivered in order requested:  $O_2$ ,  $O_3$ ,  $O_4$  wait behind  $O_1$*

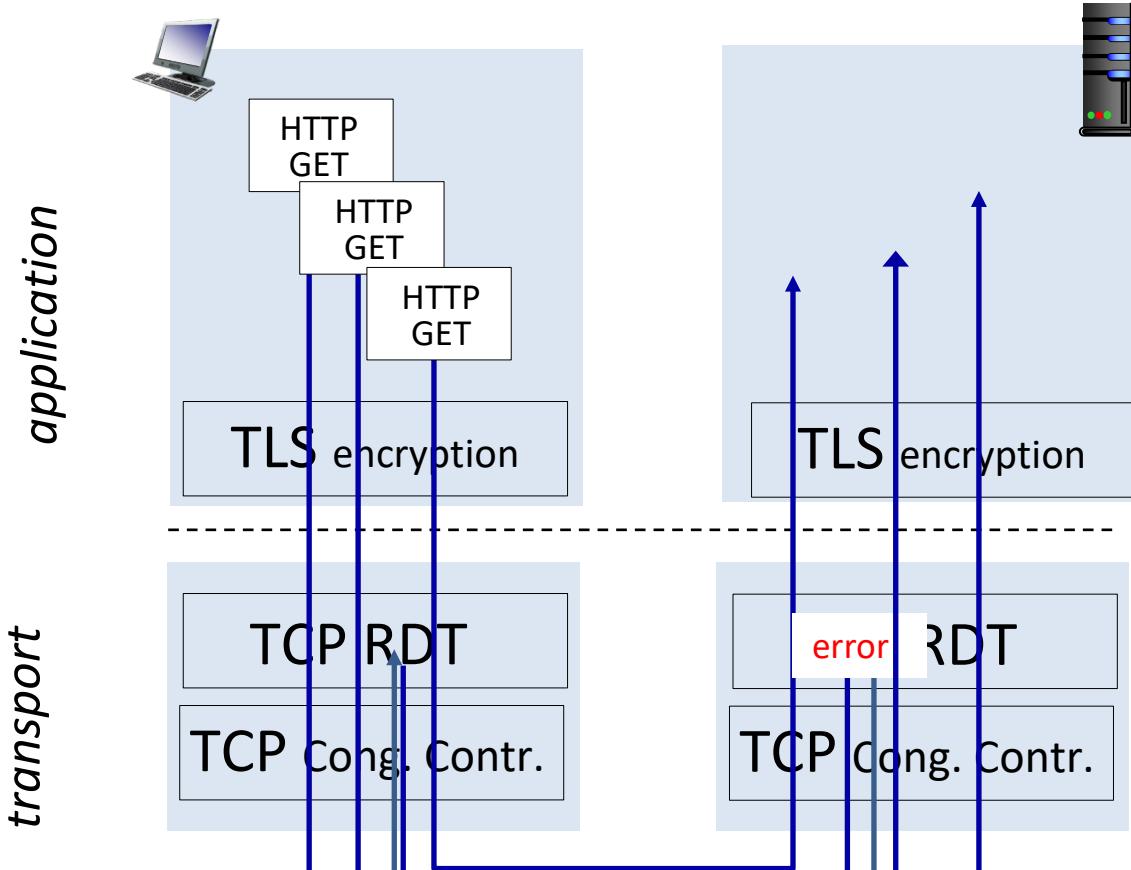
# Recall improvement: HTTP/2 : mitigating HOL blocking

HTTP/2 (RFC 7540 ): objects divided into frames, frame transmission interleaved



# QUIC: streams: with parallelism against HOL blocking

multiple application-level “streams” multiplexed over single QUIC connection

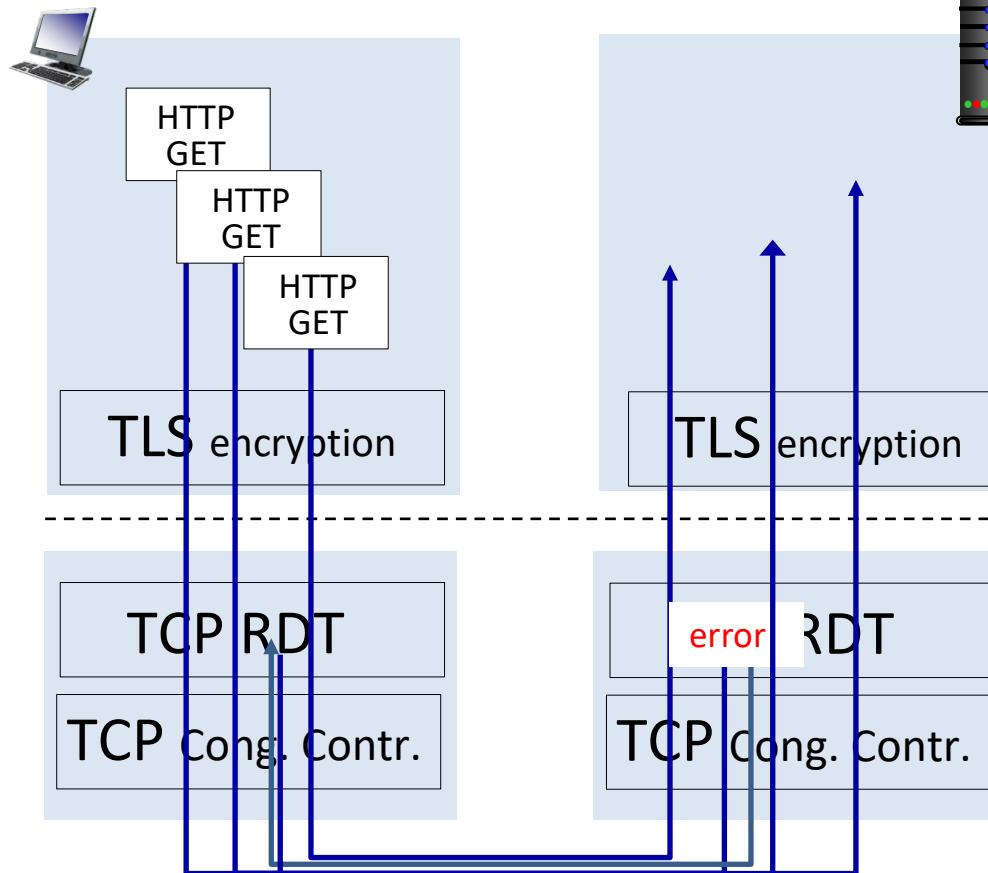


(a) HTTP 1.1

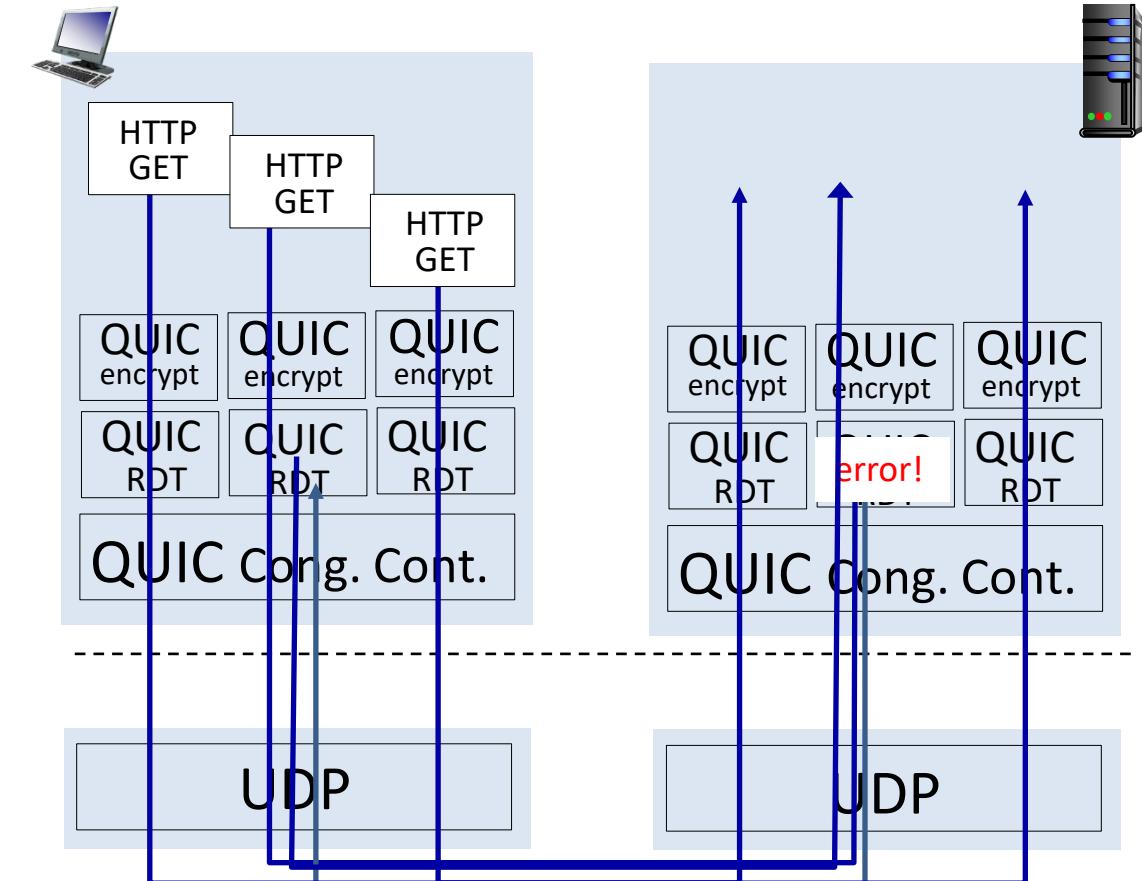
# QUIC: streams: parallelism, no HOL blocking

multiple application-level “streams” multiplexed over single QUIC connection

application  
transport



(a) HTTP 1.1



(b) HTTP/2 with QUIC: no HOL blocking

# Roadmap -- Internet evolution @edge



## P2P apps and overlays for info sharing

- Collaborate/form-overlays to *find* content:
- Collaborate/form-overlays to *fetch* content

## Media Streaming

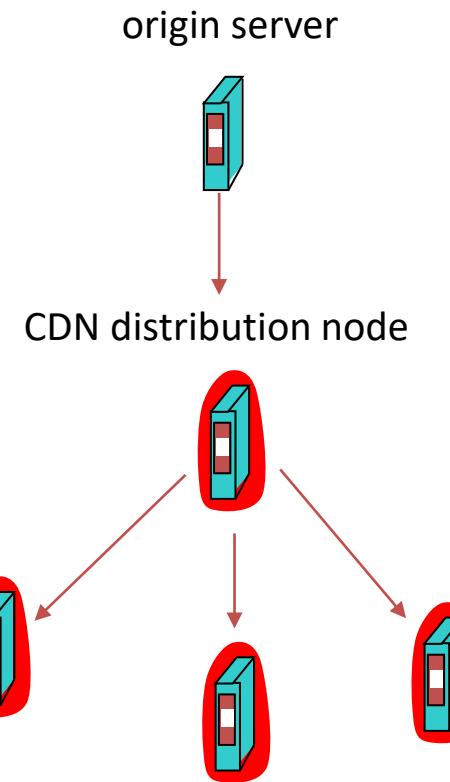
- Application classes, challenges
- Today's applications representative technology
  - Recovery from jitter
  - Streaming protocols and new proposals

## ● Parallel with p2p content sharing: CDN: overlays infrastructure for content delivery

# Content distribution networks (CDNs)

- Business side of the content sharing/distribution idea, building on Distributed Systems research
- **Scalability** big problem to stream large files from single origin server in real time to millions end-hosts
- A solution: Content replication at more servers
  - content downloaded to CDN servers in *pull or push manner*
  - content “close” to user avoids impairments (loss, delay) of long paths

Notice the “parallel” with **overlay networks** in P2P applications!



# Content distribution Networks (CDNs)

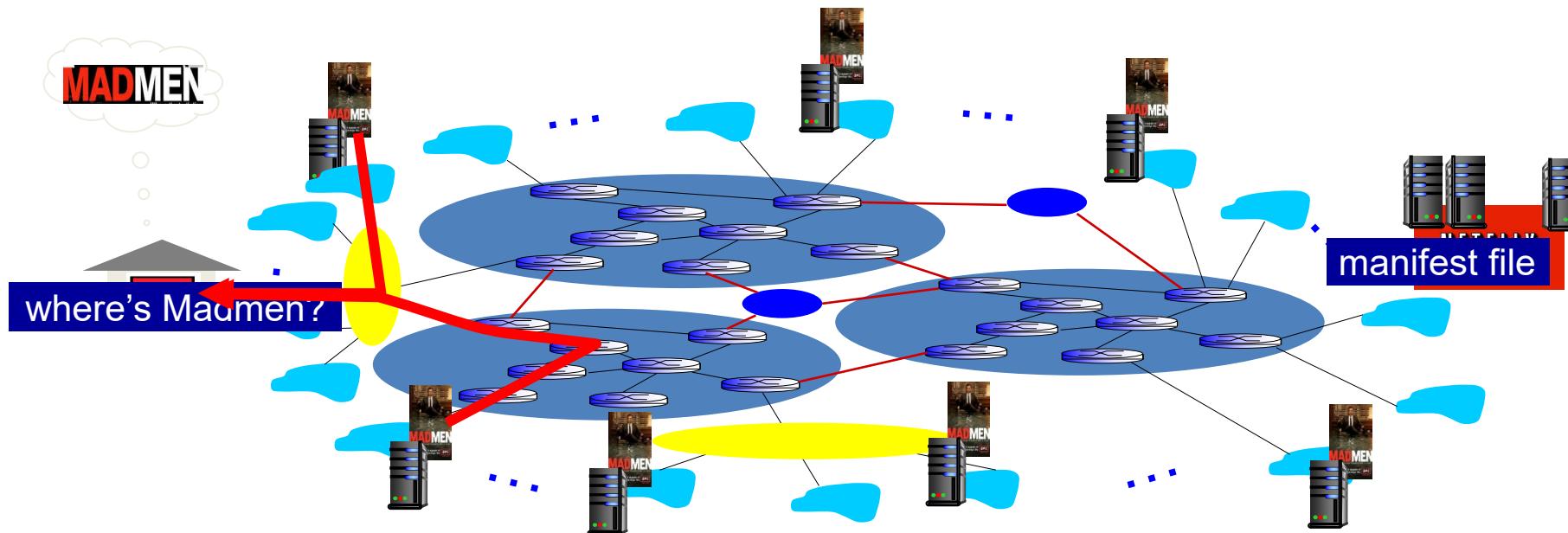
---

Q: How to store/serve multiple copies of videos at multiple geographically distributed sites (*CDN*)

- *enter deep*: push CDN servers deep into many access networks
  - close to users
  - Akamai: 300,000 servers deployed in more than 130 countries (2021) - supports also p2p functionality
- *bring home*: smaller number (10's) of larger clusters in POP/IXPs near (but not within) access networks
  - used by former Limelight/now Edgio
- Google, Netflix use combinations of those with their private networks

# Content Distribution Networks (CDNs)

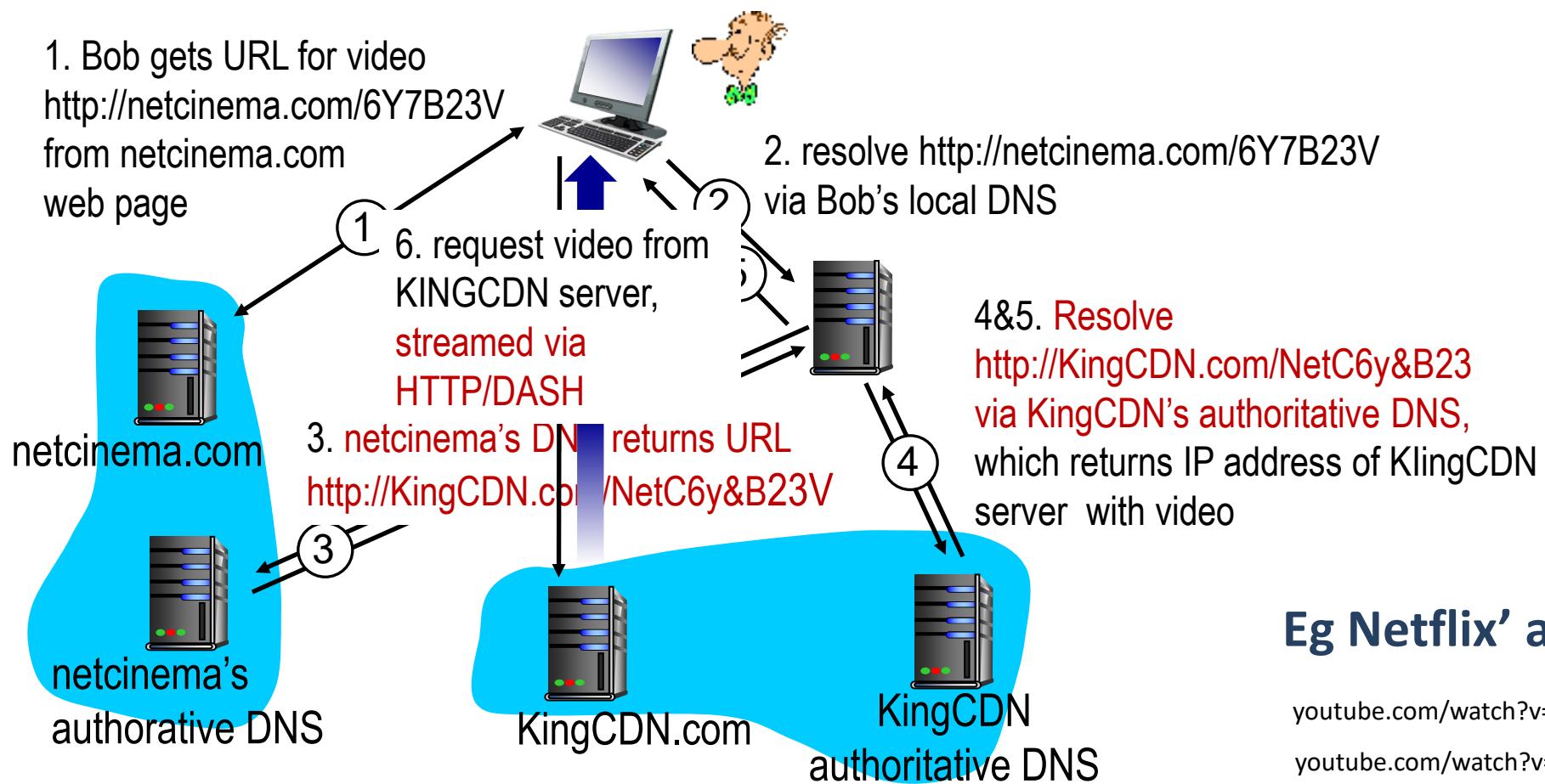
- CDN: stores copies of content at CDN nodes
  - e.g. Netflix stores copies of MadMen
- subscriber requests content from CDN
  - directed to nearby copy, retrieves content
  - **may** choose different copy *if network path congested*



# CDN: “simple” content access scenario

Bob (client) requests video <http://netcinema.com/6Y7B23V>

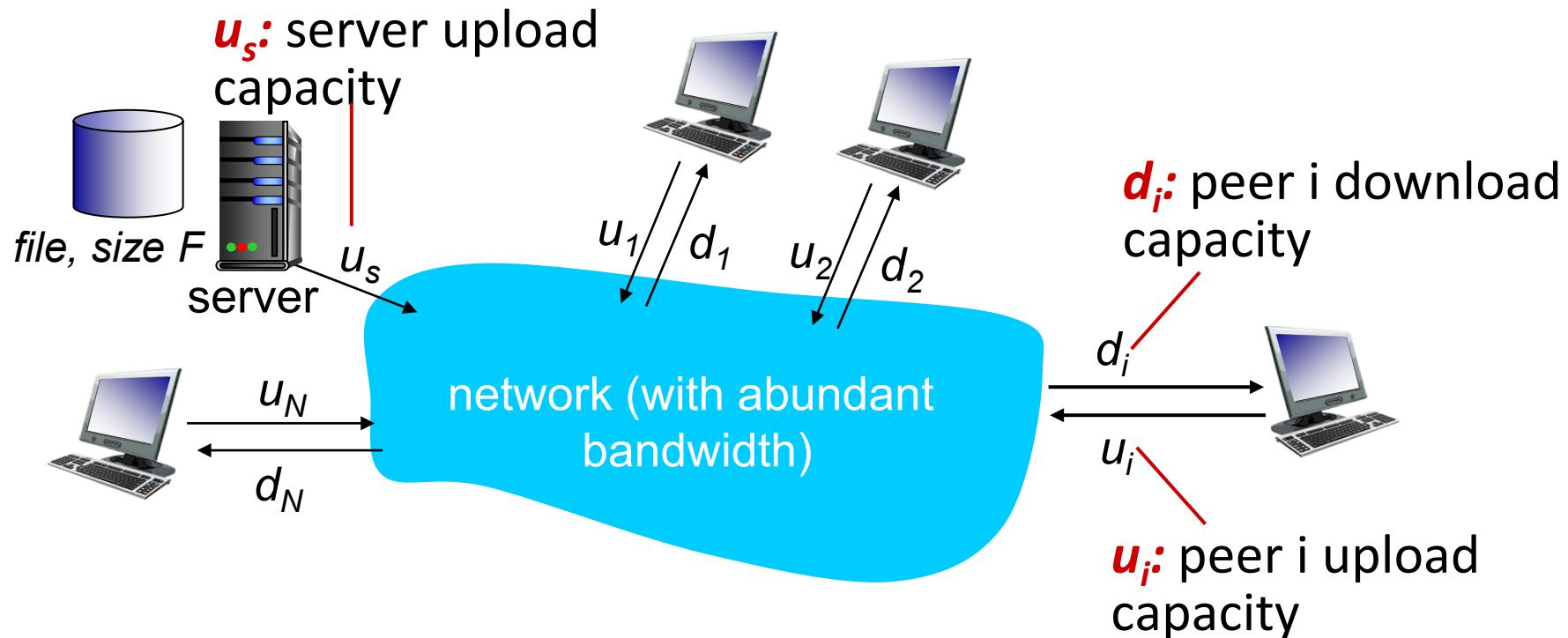
- video stored in CDN at <http://KingCDN.com/NetC6y&B23V>



# On performance of file distribution: from-single-server vs distributed /P2P

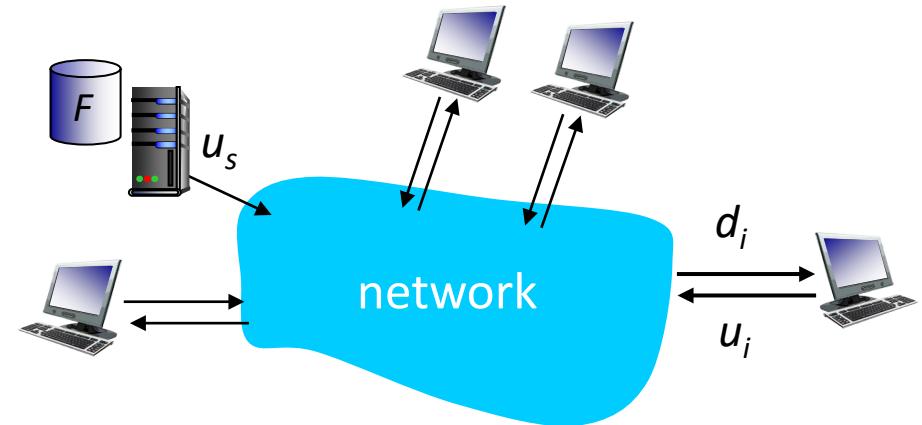
**Q:** how much time to distribute file (size  $F$ ) from one server to  $N$  peers?

- peer upload/download capacity is limited resource



# File distribution time: from a single server

- *server transmission*: must sequentially send (upload)  $N$  file copies:
  - time to send one copy:  $F/u_s$
  - time to send  $N$  copies:  $NF/u_s$
- *client*: each client must download file copy
  - $d_{min}$  = min client download rate
  - min client download time:  $F/d_{min}$



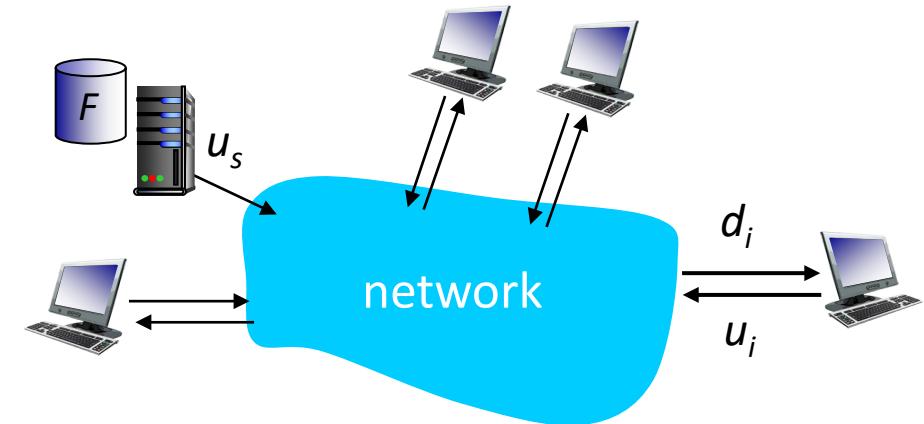
*time to distribute  $F$   
to  $N$  clients using  
client-server approach*

$$D_{c-s} \geq \max\{NF/u_s, F/d_{min}\}$$

increases linearly in  $N$

# File distribution time: P2P

- *server transmission*: must upload at least one copy:
  - time to send one copy:  $F/u_s$
- *client*: each client must download file copy
  - min client download time:  $F/d_{min}$
- *clients*: as aggregate must download  $NF$  bits
  - max upload rate (limiting max download rate) is  $u_s + \sum u_i$



time to distribute  $F$   
to  $N$  clients using  
P2P approach

$$D_{P2P} \geq \max\{F/u_s, F/d_{min}, NF/(u_s + \sum u_i)\}$$

increases linearly in  $N$  ...  
... but so does this, as each peer brings service capacity

## Up to this point summary of evolving internet protocols for supporting evolving needs of apps

- use **UDP** to avoid TCP congestion control (delays) for time-sensitive traffic; or **multiple TCP** connections (DASH + new http/2/3 +quic, ....)
  - Buffering and client-side **adaptive playout delay**: to compensate for delay
- **CDN**: bring content closer to clients
- server side **matches stream bandwidth** to available client-to-server path bandwidth (DASH)
  - chose among pre-encoded stream rates
  - dynamic server-encoding rate

Coming up: how is the **Network Core** evolving to support needs of evolving types of traffic? What about **Security** needs?



# Reading instructions and pointers for further study

P2P apps and overlays for info sharing Ch: 2.5

Media Streaming & support from applications (in connection to transport layer updates:  
2.6, 3.8 2

- CDN and some of the science behind an example such NW, ie Tom Leighton's work <http://people.csail.mit.edu/ftl/index.html> <https://dblp.org/pid/l/FrankThomsonLeighton.html> [https://en.wikipedia.org/wiki/Akamai\\_Technologies](https://en.wikipedia.org/wiki/Akamai_Technologies)
- Cf: Cardwell, Neal, et al. "BBR congestion-based congestion control." Communications ACM 2017 (BBR (Bottleneck Bandwidth and Round-trip propagation time): deployed on Google's (internal) backbone network) <http://dx.doi.org/10.1145/3009824>
- Eddie Kohler, Mark Handley, and Sally Floyd. 2006. Designing DCCP: Congestion control without reliability. *SIGCOMM Comput. Commun. Rev.* 36, 4 (August 2006), 27-38. DOI=10.1145/1151659.1159918 <http://doi.acm.org/10.1145/1151659.1159918>
- Langley et-al; Quic; ACM SIGCOMM '17, DOI: <https://doi.org/10.1145/3098822.3098842>
- Jussi Kangasharju, James Roberts, Keith W. Ross, Object replication strategies in content distribution networks, Computer Communications, Volume 25, Issue 4, 1 March 2002, Pages 376-383, ISSN 0140-3664, [http://dx.doi.org/10.1016/S0140-3664\(01\)00409-1](http://dx.doi.org/10.1016/S0140-3664(01)00409-1) .
- K.L Johnson, J.F Carr, M.S Day, M.F Kaashoek, The measured performance of content distribution networks, Computer Communications, Volume 24, Issue 2, 1 February 2001, Pages 202-206, ISSN 0140-3664, [http://dx.doi.org/10.1016/S0140-3664\(00\)00315-7](http://dx.doi.org/10.1016/S0140-3664(00)00315-7) .
- *p2p sharing, dissemination and media streaming*
  - J. Mundinger, R. R. Weber and G. Weiss. Optimal Scheduling of Peer-to-Peer File Dissemination. *Journal of Scheduling*, Volume 11, Issue 2, 2008. [[arXiv](#)] [[JoS](#)]
  - Christos Gkantsidis and Pablo Rodriguez, Network Coding for Large Scale Content Distribution, in *IEEE INFOCOM*, March 2005 (Avalanche swarming: combining p2p + media streaming)

# Extra notes / for further study

---

# Distributed Hash Tables (DHT)

## Implementation:

- Hash function maps entries to nodes (**insert**)
- Node-overlay has *structure* (Distributed Hash Table ie a distributed data structure, eg. Ring, Tree, cube) using it, do:
  - **Lookup/search**: find the node responsible for item; that one knows where the item is

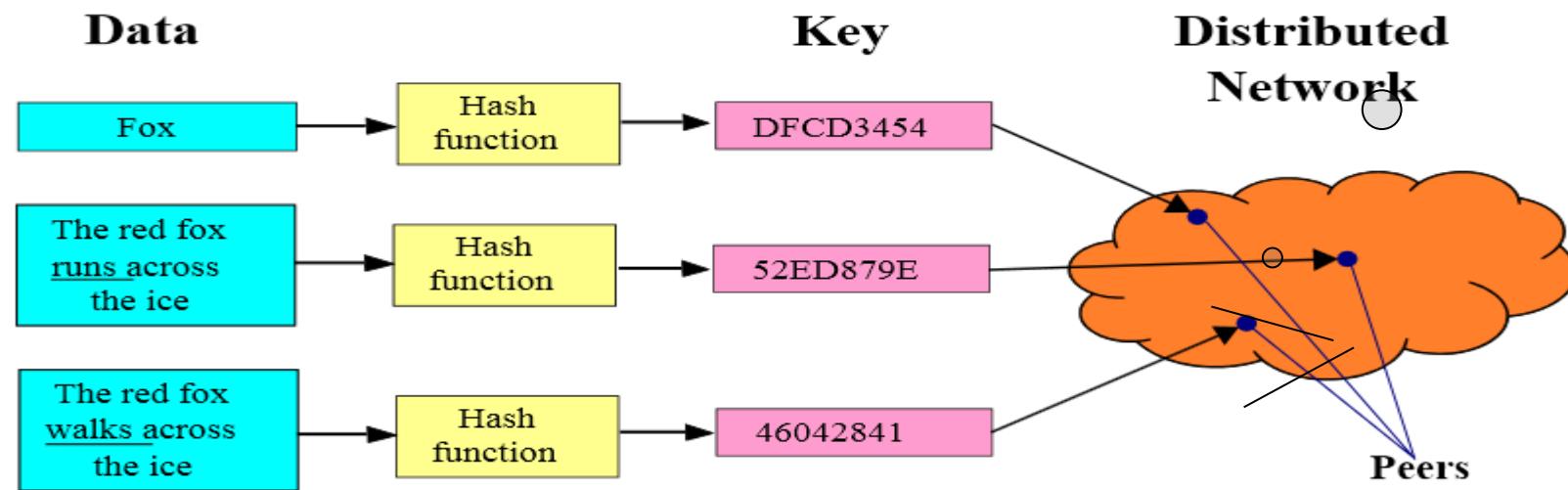
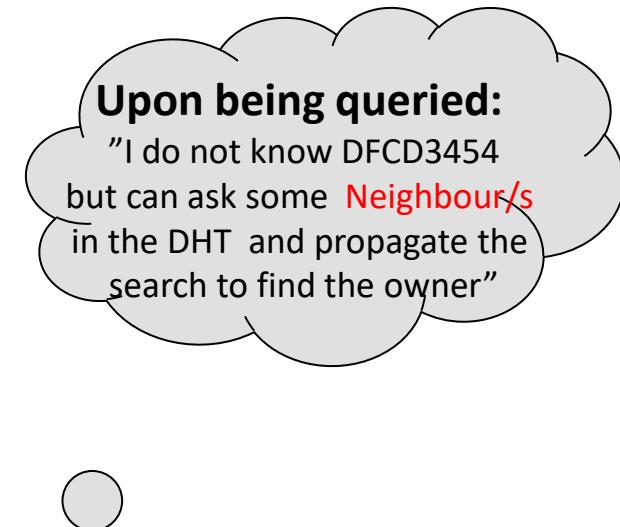


figure source: wikipedia

# (Recalling hash tables)

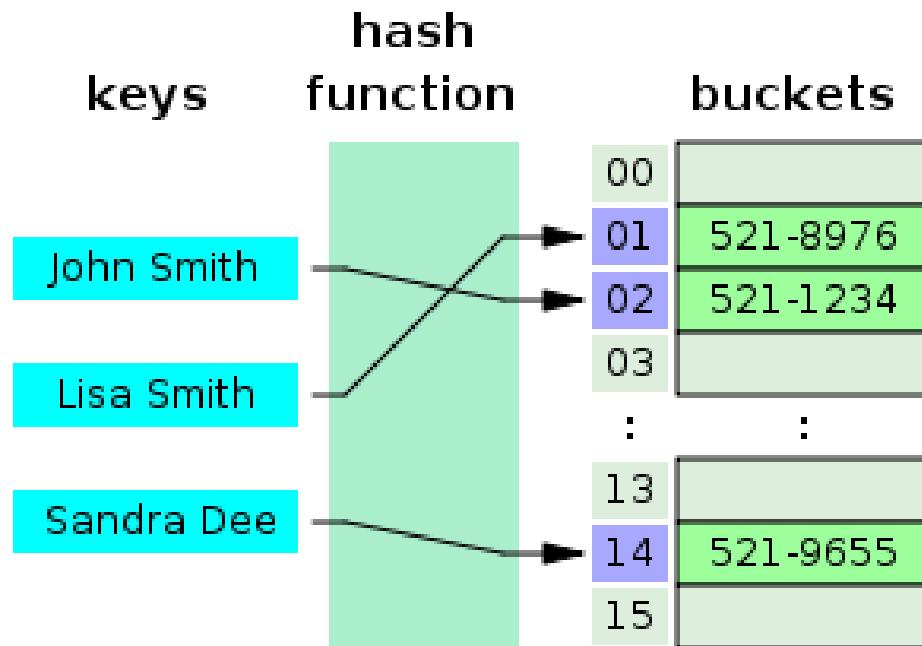
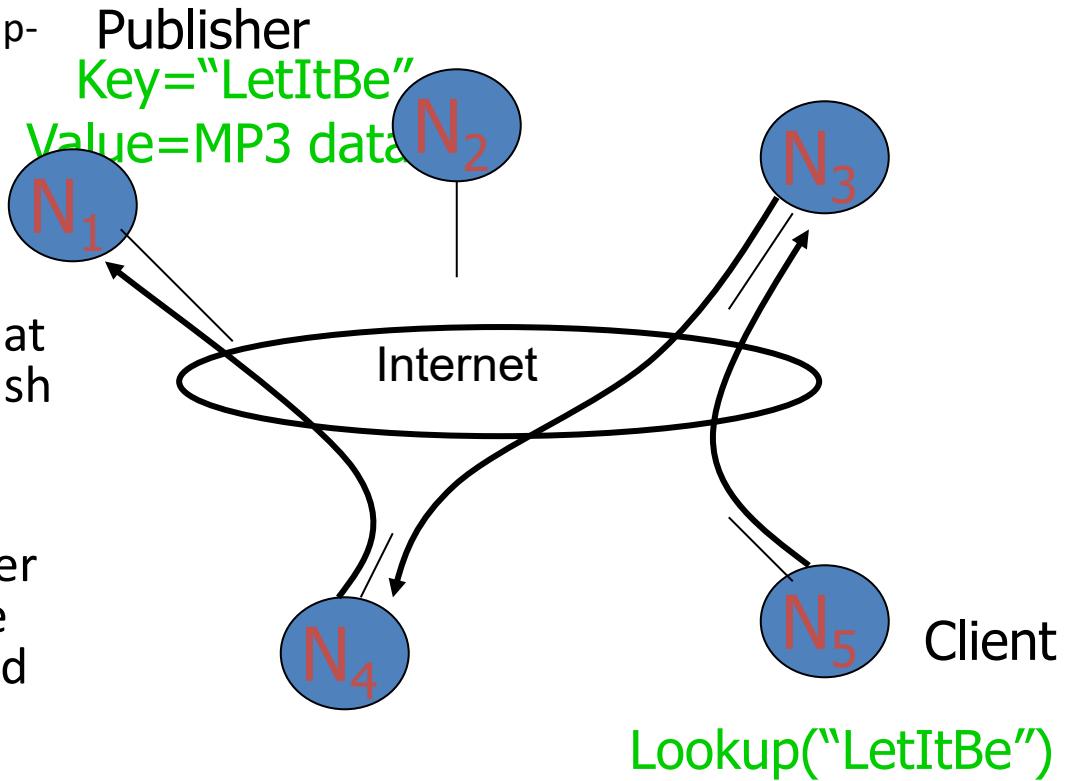


figure source: wikipedia; "Hash table 3 1 1 0 1 0 0 SP" by Jorge Stolfi - Own work. Licensed under CC BY-SA 3.0 via Commons - [https://commons.wikimedia.org/wiki/File:Hash\\_table\\_3\\_1\\_1\\_0\\_1\\_0\\_0\\_SP.svg#/media/File:Hash\\_table\\_3\\_1\\_1\\_0\\_1\\_0\\_0\\_SP.svg](https://commons.wikimedia.org/wiki/File:Hash_table_3_1_1_0_1_0_0_SP.svg#/media/File:Hash_table_3_1_1_0_1_0_0_SP.svg)

# Distributed-Hash-Table-Based p2p sharing

- **Join:**
  - get connected in the overlay through info from bootstrap-node & using the specific DHT algorithm (eg Chord)
  - Start maintaining of files that you are responsible for (following the hash function)
  - NOTE: upon leaving DHT needs restructuring!
- **Publish:** tell which files you have, to the peers that will be responsible for them (according to the hash function)
- **Search:** ask *the appropriate neighbour*, who either is responsible for the searched file or will ask the next appropriate neighbor, and so on; guaranteed search time; commonly in  $O(\log \text{Nodes})$
- **Fetch:** get the file directly from peer

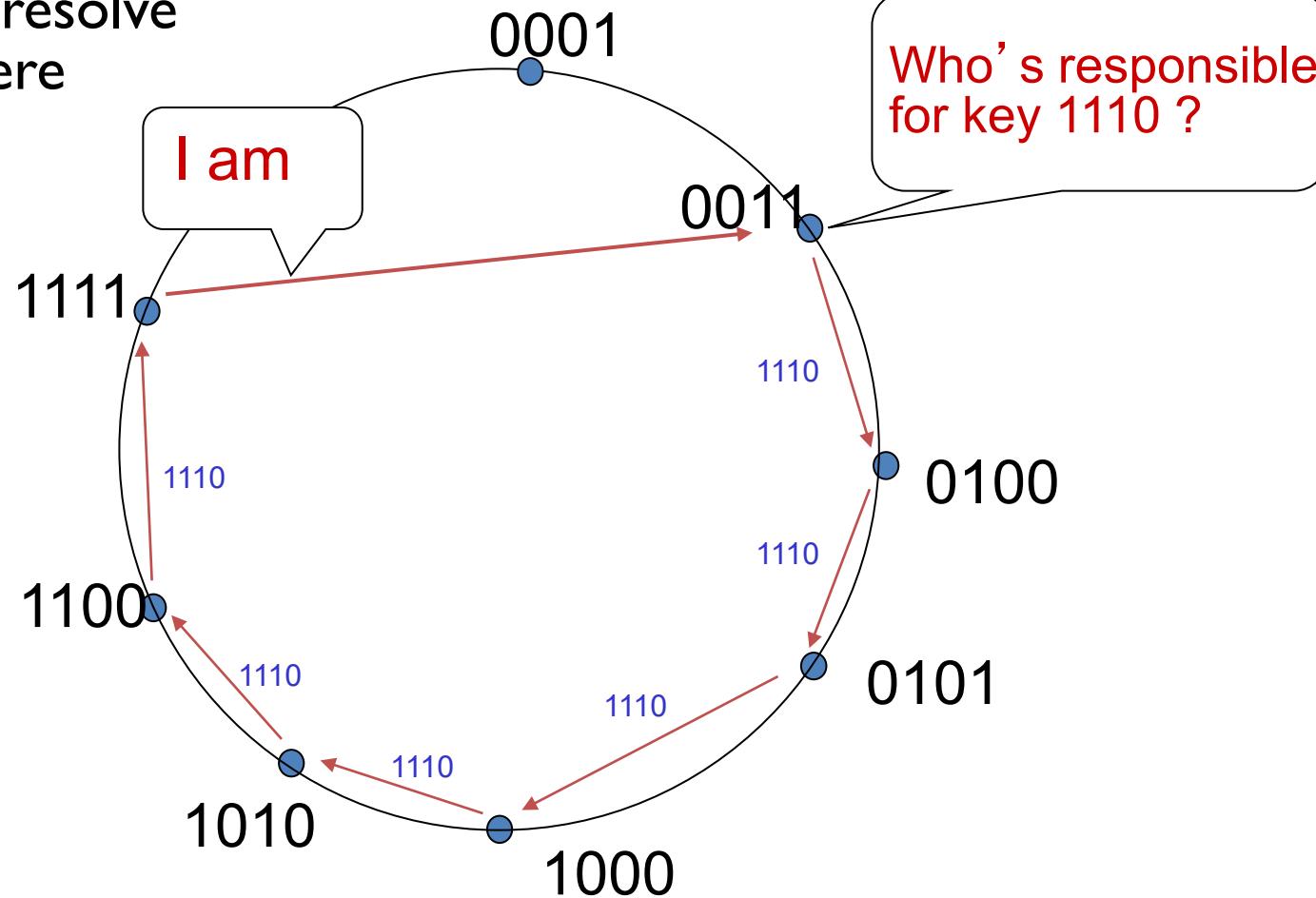


Challenges [cf related literature@end of notes]:

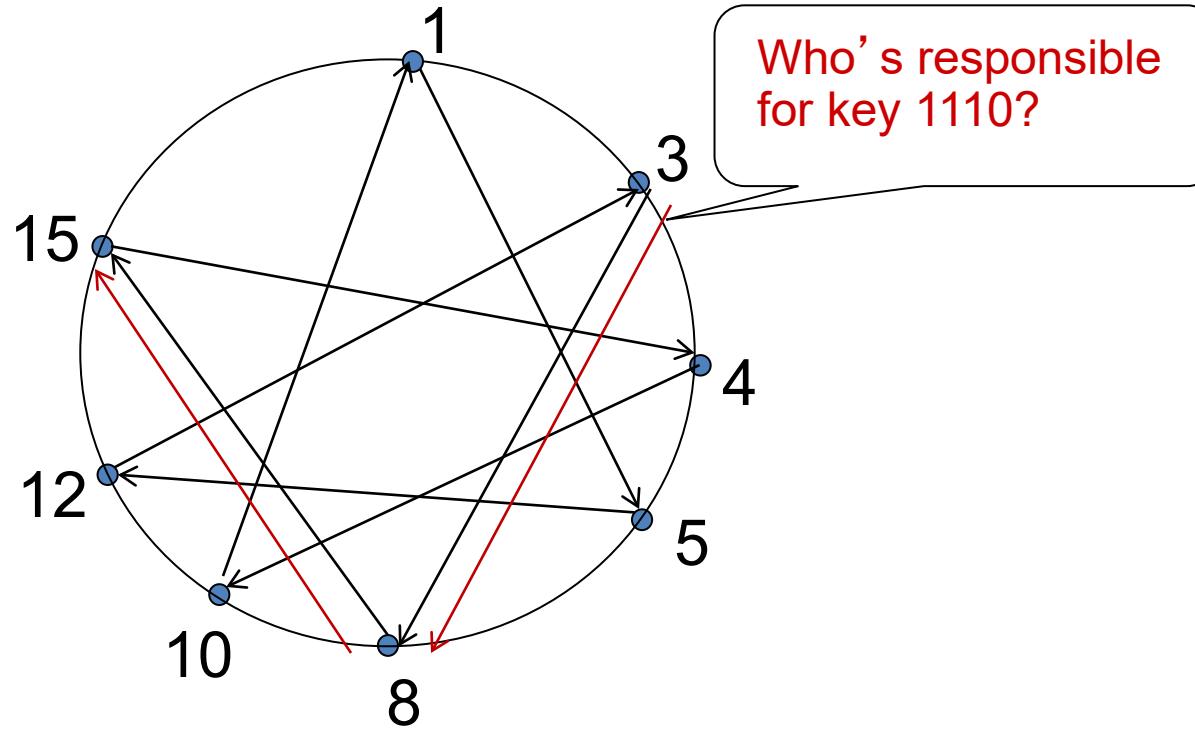
- Keep the hop count (**asking chain**) small
- Keep the routing tables (**#neighbours**) “right size”
- Stay robust despite rapid changes in membership (churn)

# e.g. Circular DHT (I)

$O(N)$  messages  
on average to resolve  
query, when there  
are  $N$  peers



# Circular DHT with shortcuts

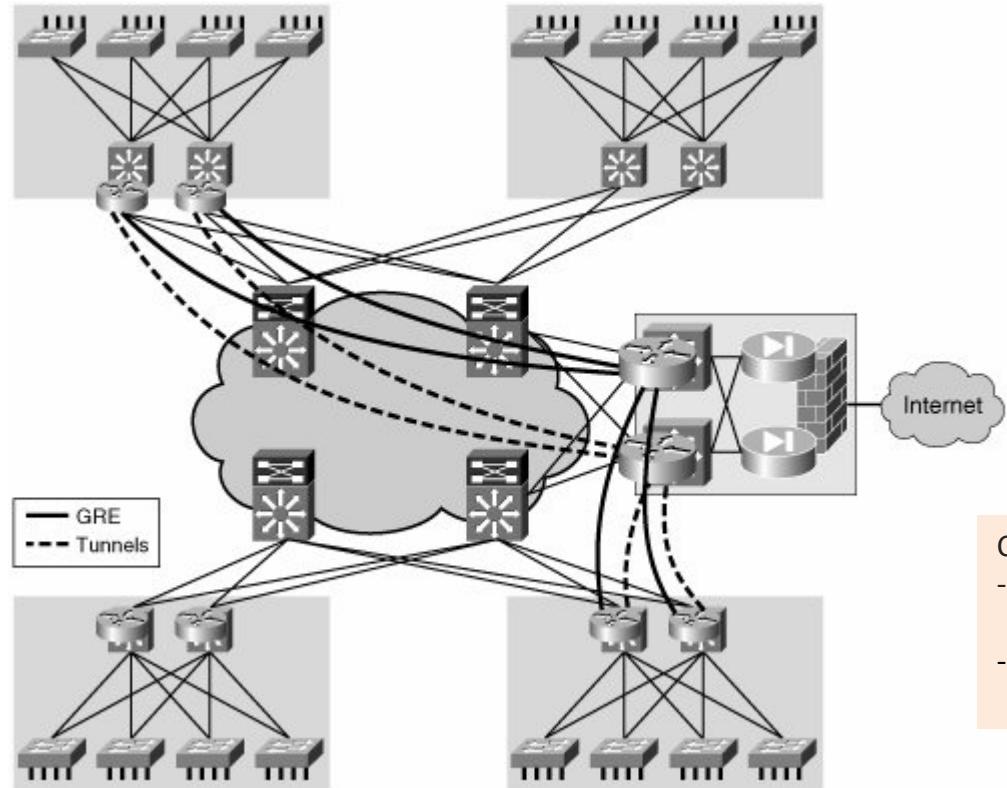


- Here: reduced from 6 to 2 messages.
- possible to design shortcuts so  $O(\log N)$  neighbors,  $O(\log N)$  messages in query

# Router Overlays – in support of Software Defined Networks

for e.g.

- distributing responsibility of control and routing (5G)
- protection/mitigation of flooding attacks, collaborate for filtering flooding packets

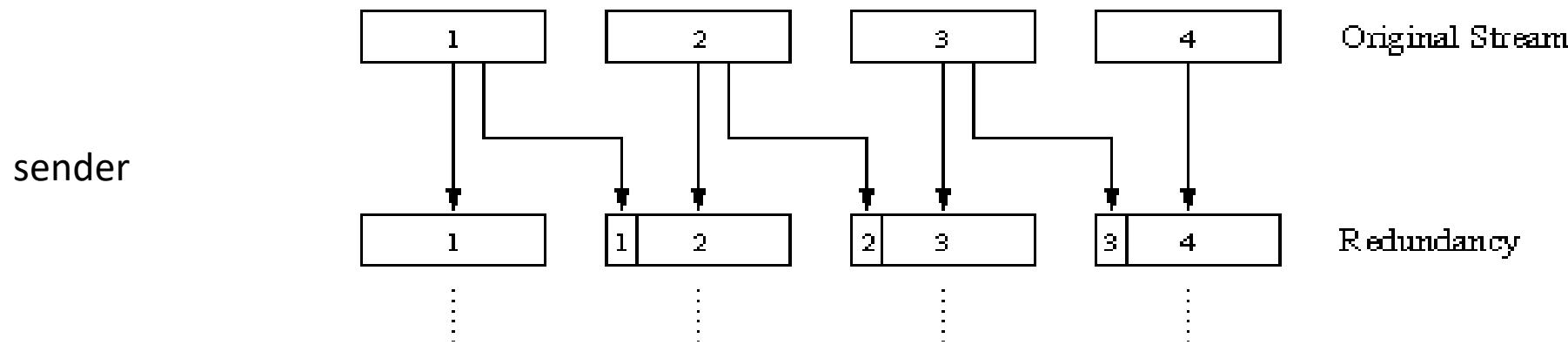


Cf eg:

- Fu, Z., et al. Off the Wall: Lightweight Distributed Filtering to Mitigate Distributed Denial of Service Attacks. IEEE SRDS 2012.
- Gulisano, V. et al. "STONE: A streaming DDoS defense framework." Expert Systems with Applications 42.24 (2015): 9620-9633.

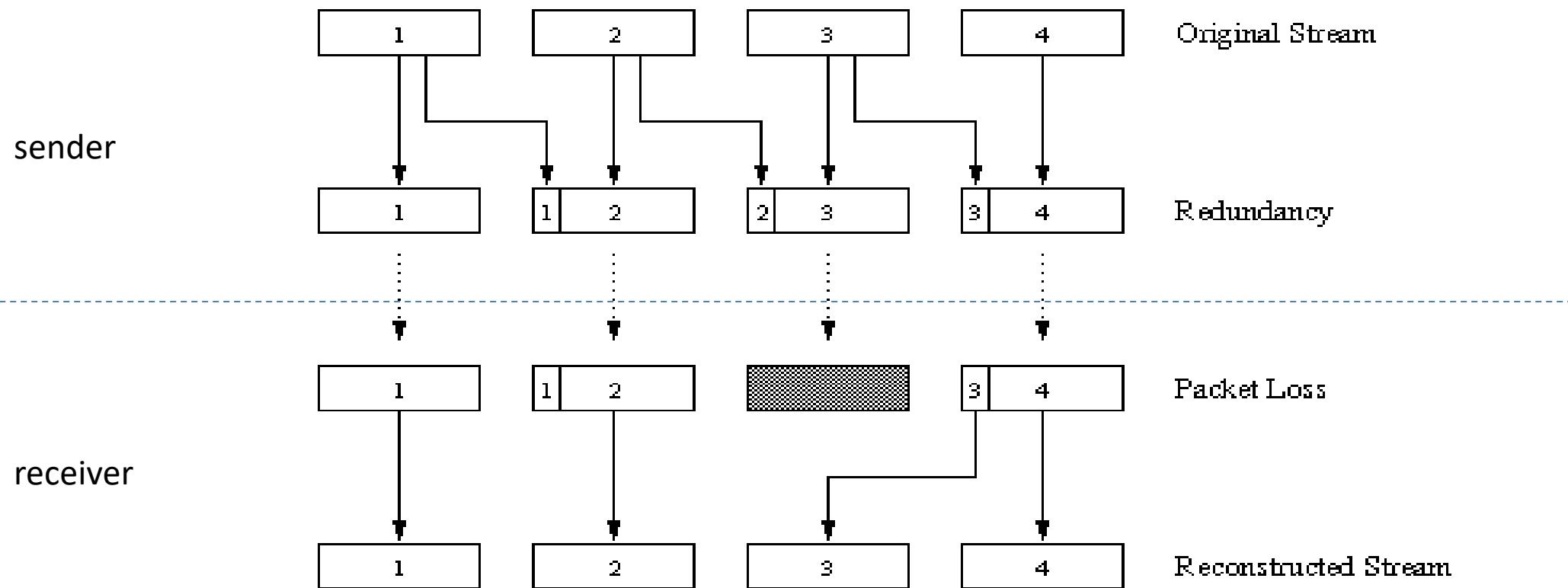
# Recovery From Packet Loss: Forward Error Correction (FEC)

Eg.: through piggybacking Lower Quality Stream



# Recovery From Packet Loss: Forward Error Correction (FEC)

Eg.: through piggybacking Lower Quality Stream

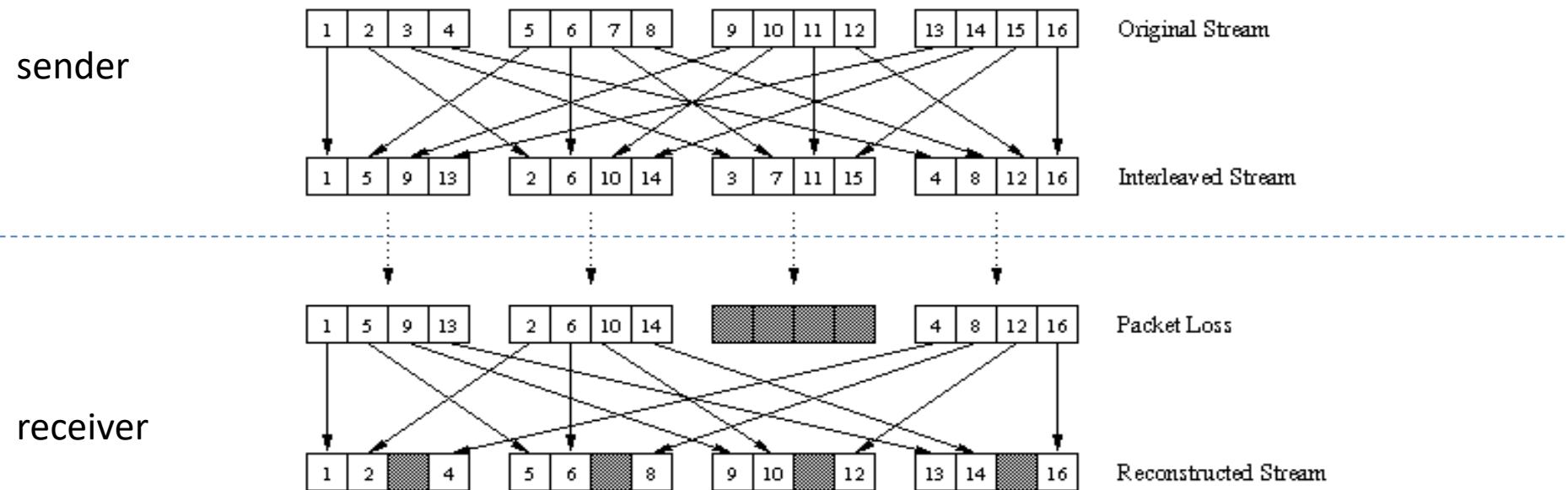


Why care about FEC?

# Recovery From Packet Loss/FEC (cont)

## Example method 2. Interleaving:

- Upon loss, have a set of partially filled chunks
- Playout time must adapt to receipt of group (risk wrt Real-Time requirements)



# Multimedia networking: application types

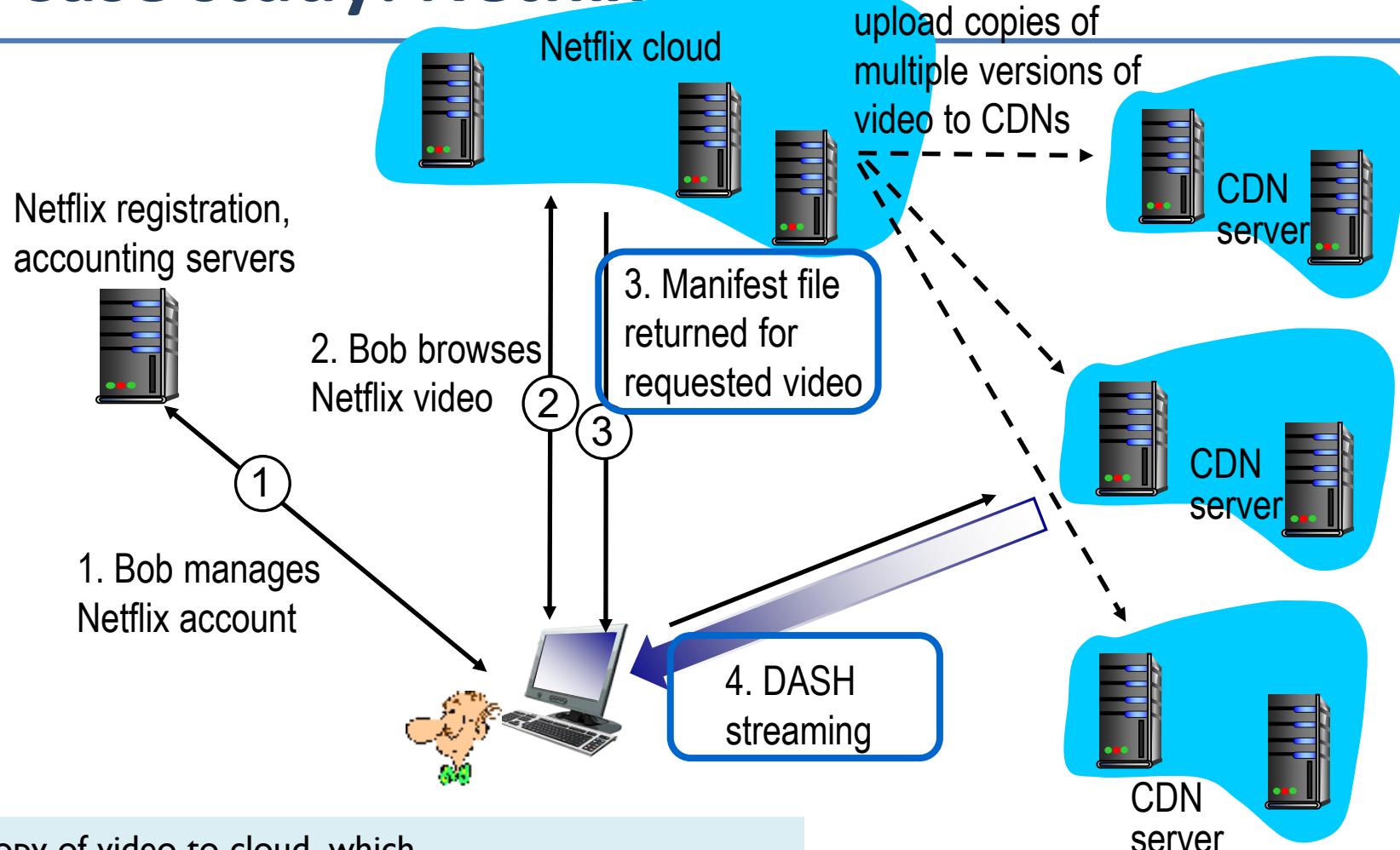
- *streaming stored* audio, video
  - can begin play before downloading entire file (implies storing/buffering at client); e.g., YouTube, Netflix,
- *streaming live* audio, video
- *conversational* voice/video
  - interactive nature limits delay tolerance; e.g., Skype

## Fundamental characteristics:

- typically **delay sensitive; care about**
  - end-to-end delay
  - delay jitter
- **loss tolerant**: infrequent losses cause only minor glitches
- In contrast to traditional data-traffic apps, which are *loss intolerant* but *delay tolerant*.

Jitter is the variability of packet delays within the same packet stream

# Case study: Netflix



- Netflix uploads copy of video to cloud, which
  - creates multiple versions of movie (different encodings) in cloud
  - uploads versions from cloud to CDN servers;
- user downloads the suitable encoding from them

[youtube.com/watch?v=LkLLpYdDINA](https://youtube.com/watch?v=LkLLpYdDINA)

[youtube.com/watch?v=tbqcsHg-Q\\_o](https://youtube.com/watch?v=tbqcsHg-Q_o)

# Content distribution networks (CDNs)



Internet host-host communication as a service

*OTT challenges:* coping with a congested Internet

- from which CDN node to retrieve content?
- viewer behavior in presence of congestion?
- what content to place in which CDN node?



# Course on Computer Communication and Networks

## Lecture 11

### Continuously evolving Internet-working (cont)

Network core

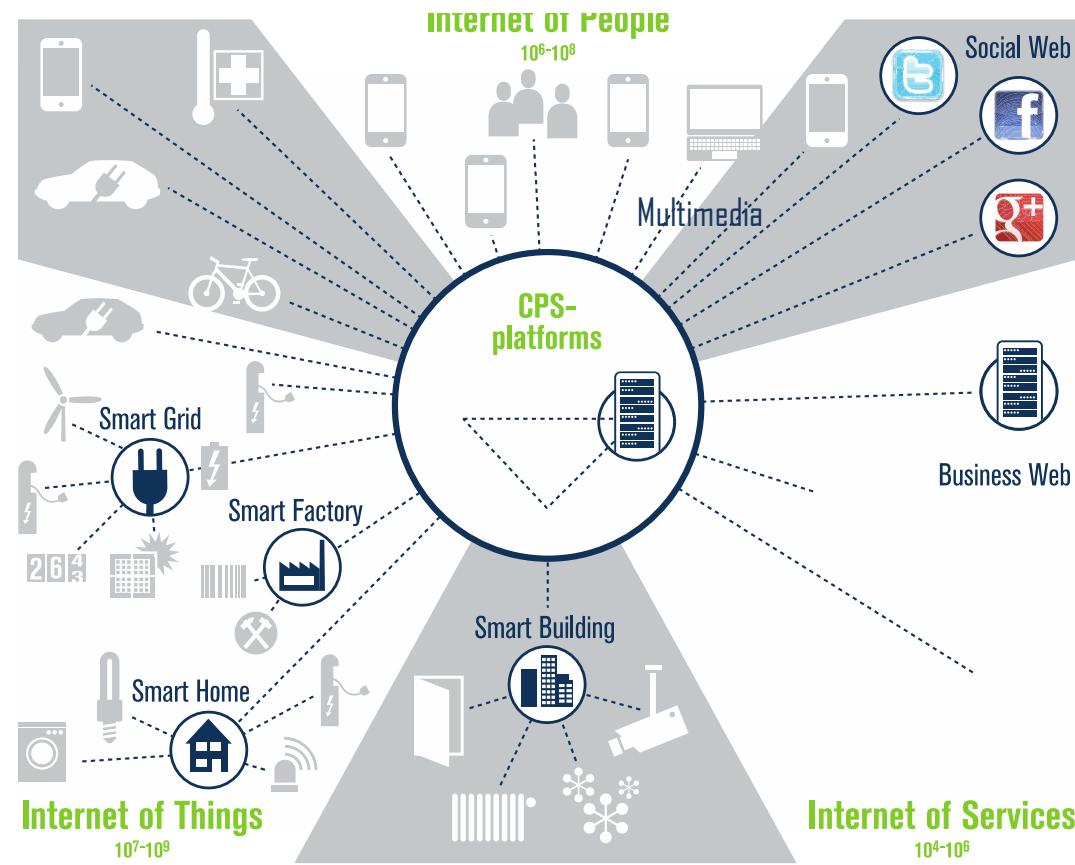
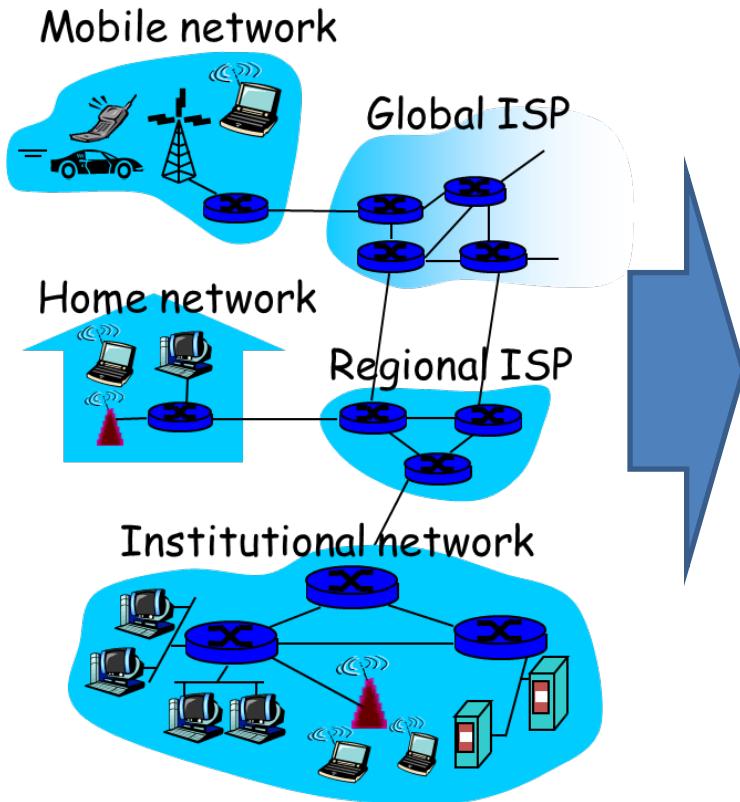
Packet Scheduling, SDN, Middleboxes, IoT implications

EDA344/DIT 423/LEU062

Lecturer: Marina Papatriantafilou

Based on the book Computer Networking: A Top Down Approach, Jim Kurose, Keith Ross, Addison-Wesley.

# Internet & its context....



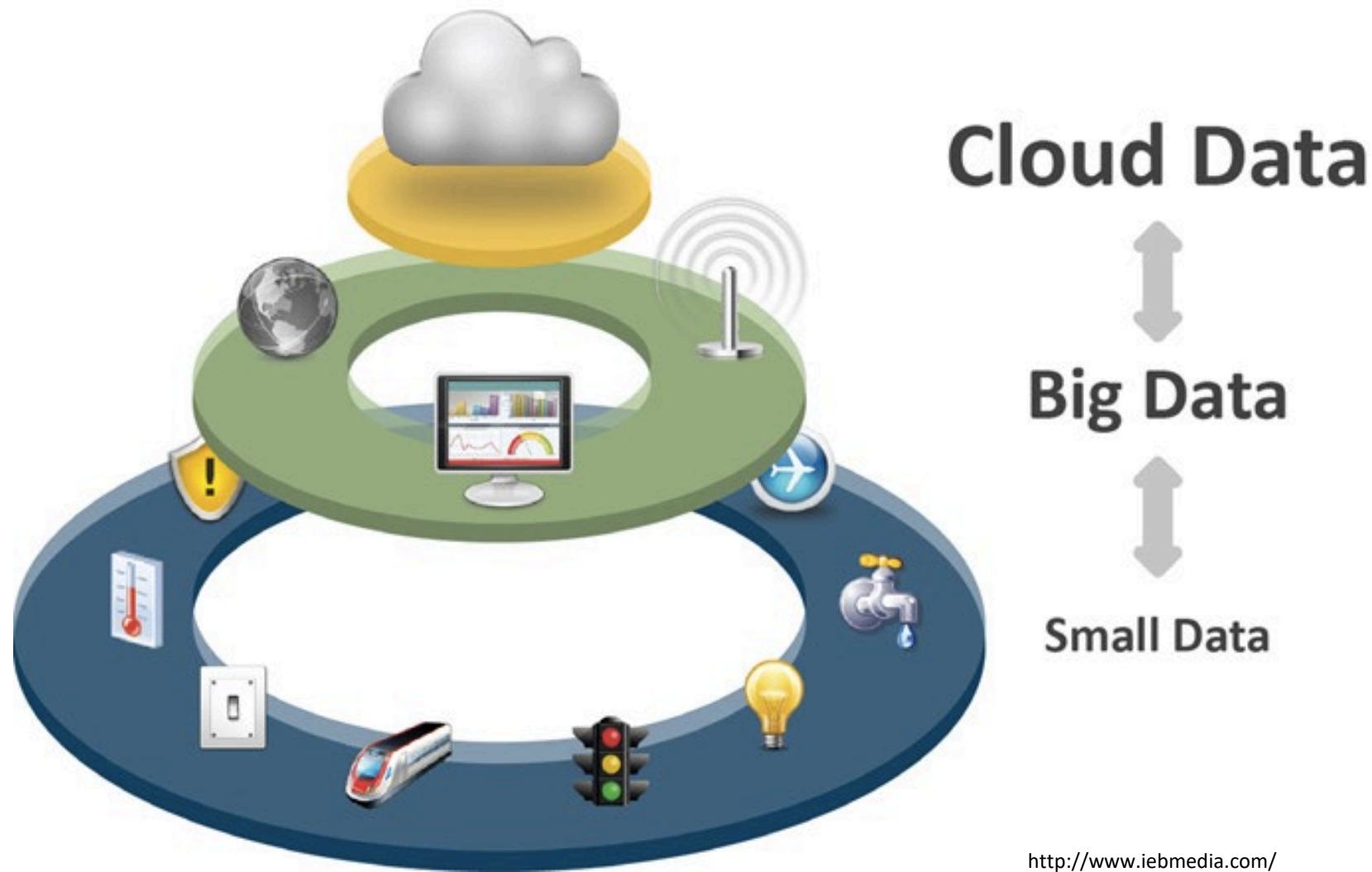
Source: Bosch Software Innovations 2012

ca year 2000: NW bandwidth & Internet - rapid growth, multimedia NW traffic and apps....:  
needs for time&bandwidth guarantees

continuous evolution ....

# IoT: Internet, Data processing and Distributed Computing in interplay

A lot of data to be communicated and processed



# Evolution of Internet layers&protocols

**Application:** protocols supporting *network applications*

http (*web*), smtp (*email*),

**virtualization&p2p, streaming, CDN, http2/3/QUIC...**

**transport:** process2process (end2end) data transfer

UDP, TCP,

**QUIC**

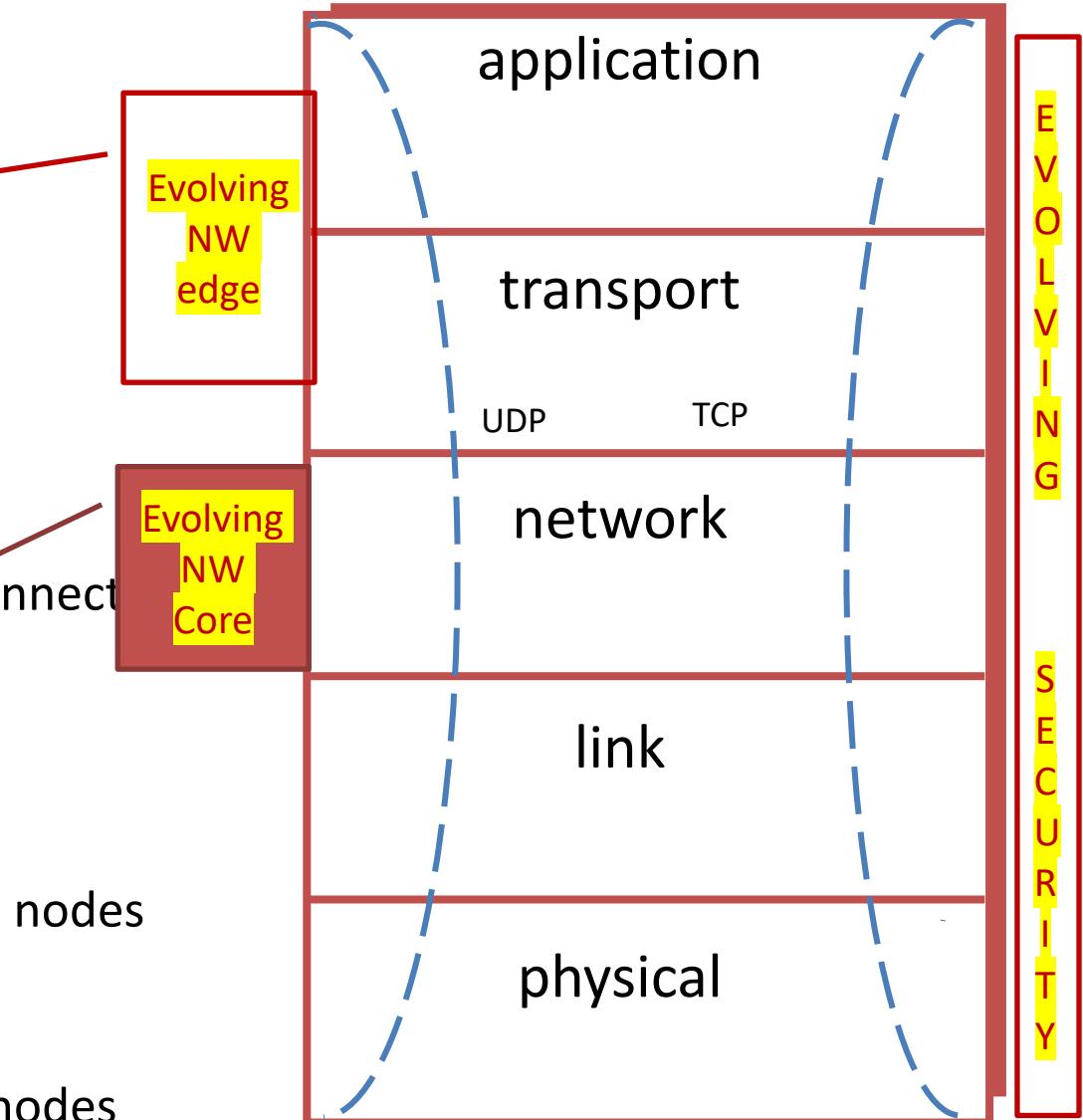
**network:** routing of datagrams (independent data-packets), connecting different physical networks

**IP addressing, routing protocols,  
virtualization, virtualization, ...**

**link:** data transfer between neighboring (physically connected) nodes

Ethernet, WiFi, ...

**physical:** bit-transmission on physical medium -- neighboring nodes



# Roadmap

**Original Internet idea:** Simple best-effort datagram service at NW layer;  
**But:** different traffic classes (e.g. multimedia) have time and bandwidth requirements  
So...



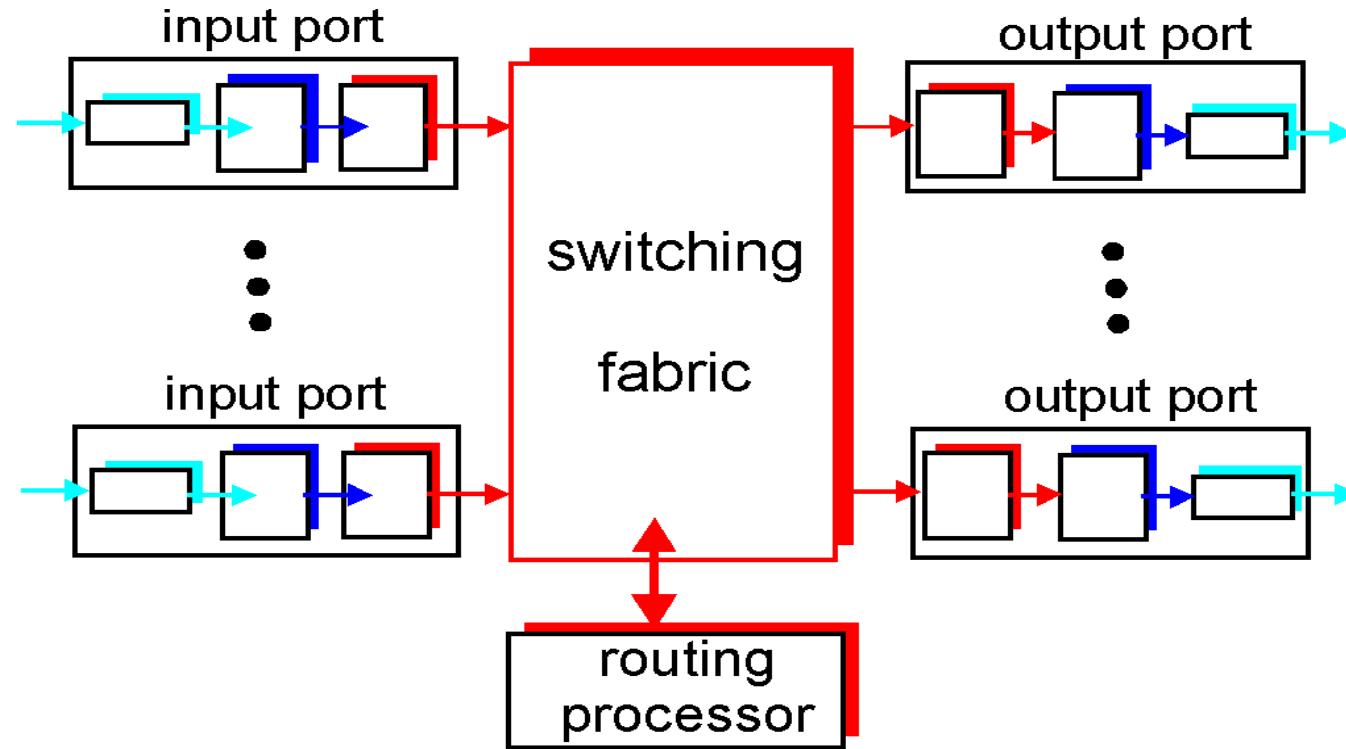
## • Zoom in (again) inside a router: [Ch. 4.2.3-4.2.5]

- **Decisions that influence timing/bandwidth guarantees** (also related with congestion-control):  
Buffer management, packet scheduling, traffic shaping

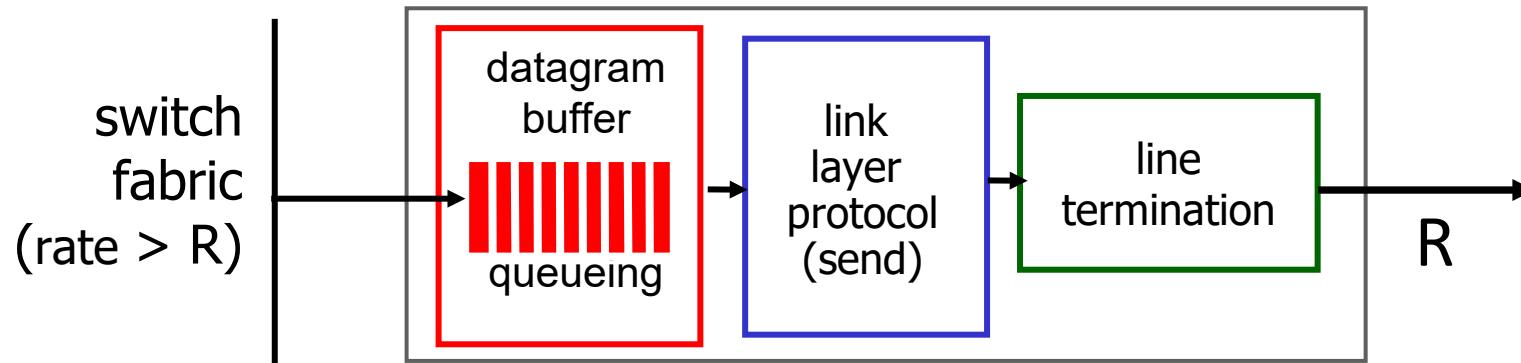
## Dealing with limitations of destination-based forwarding

- Software-Defined Networking (SDN) / NW Function Virtualization (NFV)[ch 4.4, 5.5, 4.5]
  - Concept
  - Generalized Forwarding- OpenFlow
  - Ctrl Plane – OpenFlow
- Middleboxes, "Intelligence" and controls
- Internet-of-Things in evolution: more types of traffic/devices...

# Recall: inside a router



# Recall: Output port queuing



Very large volumes of traffic here!  
Decisions with significant implications!

**A. Buffering** required when datagrams arrive from fabric faster than link transmission rate.

- **Drop policy:** which datagrams to drop if no free buffers?
- **Buffer size:** how large?

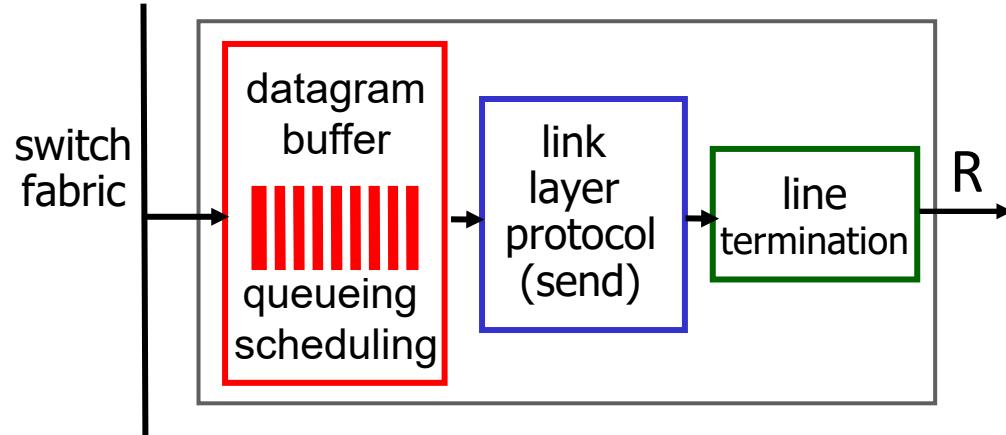
Datagrams can be lost due to congestion, lack of buffers

**B. Scheduling discipline** chooses among queued datagrams for transmission

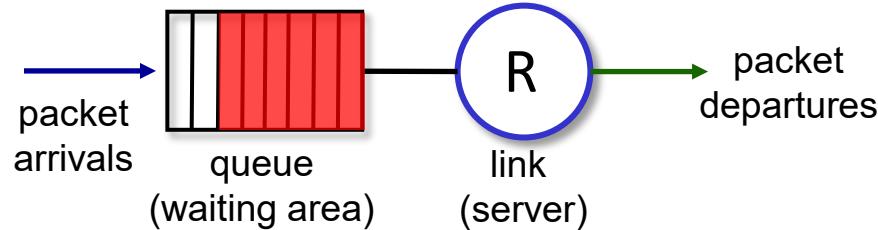
Priority scheduling – who gets best performance;  
network neutrality?

Internet service models do not specify universal standards for these

# A. Buffer Management



Abstraction: queue



**Drop-policy : when buffers are full**

- tail drop: drop arriving packet
- priority: drop/remove on priority basis
- marking: which packets to mark to **signal congestion (ECN)**

**Buffer size (B)**

- RFC 3439 rule of thumb:  
$$B = RTT * C \text{ (link capacity)}$$
  - e.g., RTT 250 msec, C = 10 Gbps,  $\rightarrow$  2.5 Gbit buffer
- more recent recommendation: with  $K$  flows,  
$$B = \frac{RTT * C}{\sqrt{K}}$$

Excessive buffering can increase delays; long RTTs: poor performance for real-time apps, sluggish TCP response

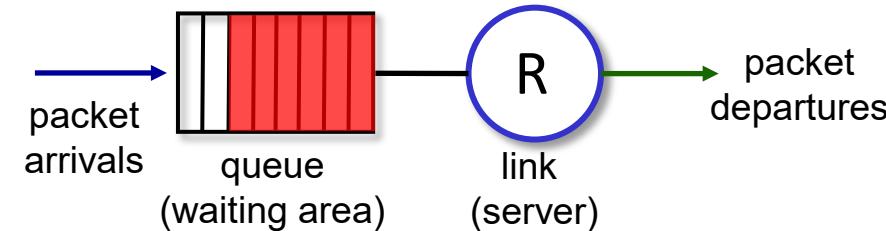
## B. Packet Scheduling

---

**packet scheduling:** deciding which packet to send next on link

- first come, first served
- priority
- round robin
- weighted fair queueing

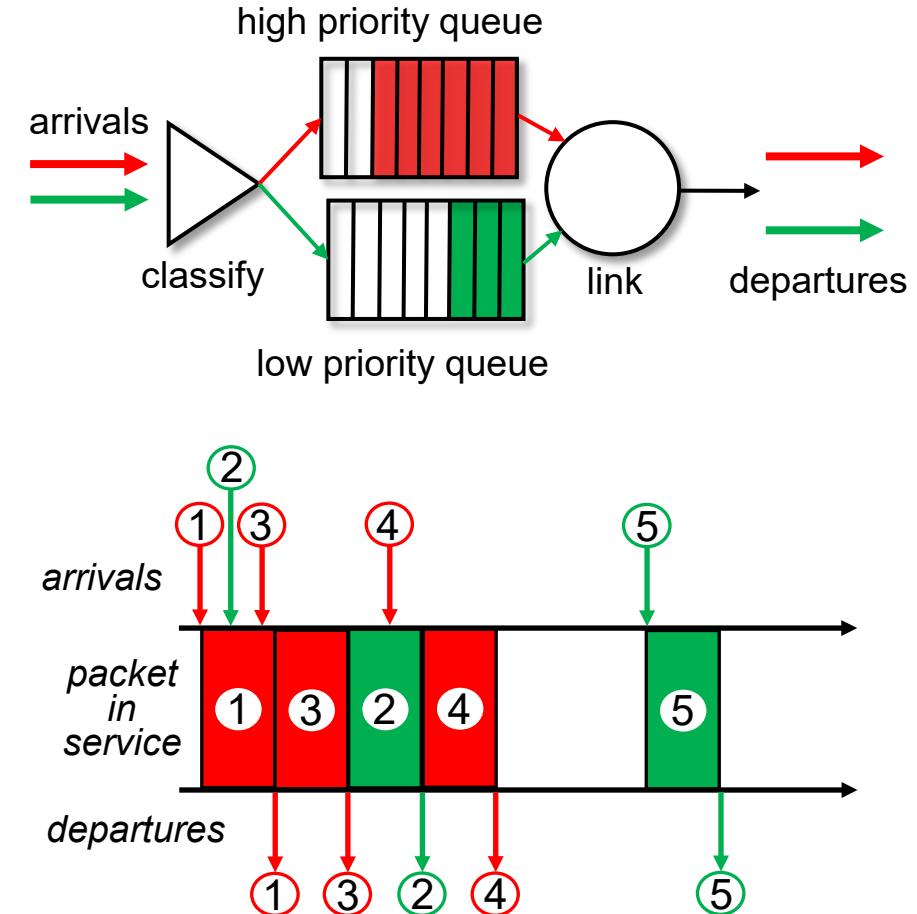
Abstraction: queue



## B. Scheduling policies: priority

### *Priority scheduling:*

- arriving traffic classified, queued by class
  - In principle any header fields could help for classification
- send packet from highest priority queue that has buffered packets



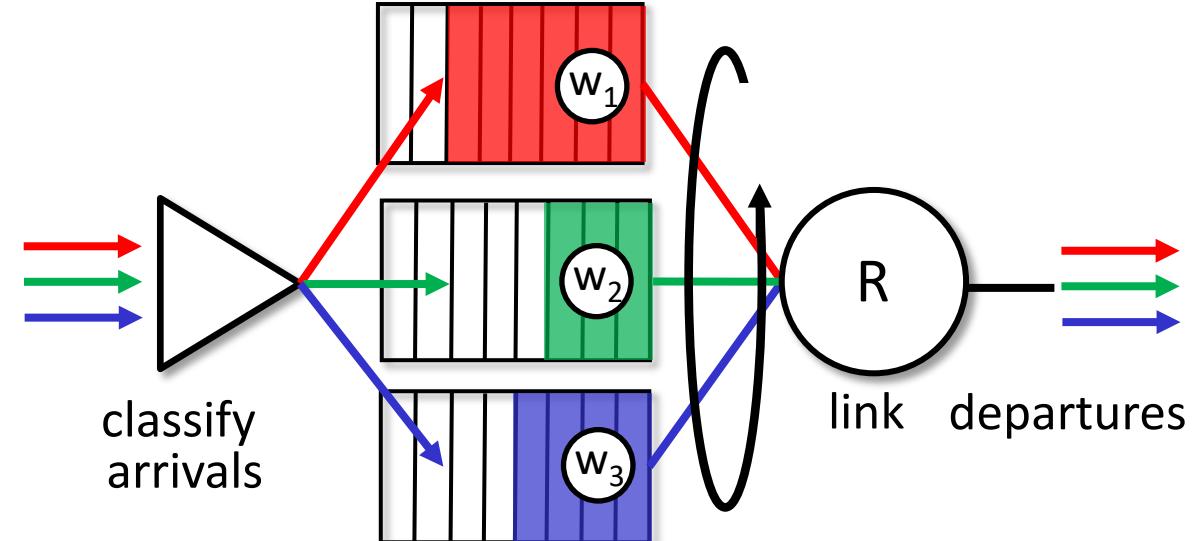
## B. Scheduling policies: weighted fair queueing

### *Weighted Fair Queuing (WFQ):*

- generalized Round Robin
- each class,  $i$ , has weight,  $w_i$ , and gets weighted amount of service in each cycle:

$$\frac{w_i}{\sum_j w_j}$$

=> minimum bandwidth guarantee (per-traffic-class)



## B. Associated concept/method:

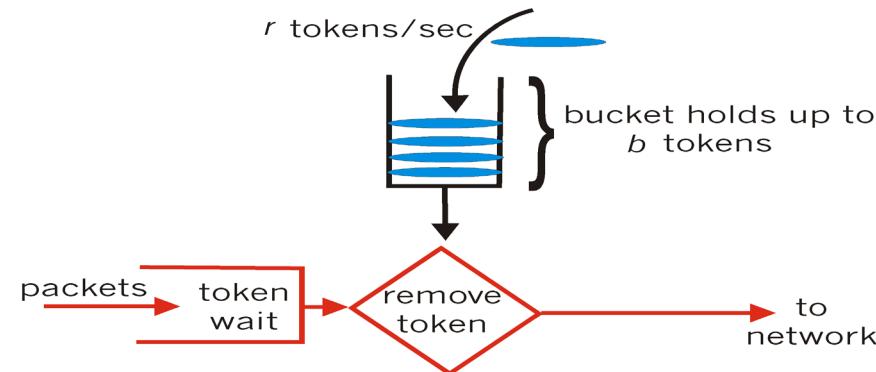
### Traffic shaping (aka policing) --Leaky Token Bucket

**Goal:** enforce agreed traffic parameters for **Quality of Service agreements (rates-latency-loss)**, limiting:

- **Average, Peak Rates** (e.g. 100 pkts/sec avg, 1500 pkts/sec peak)
- **(Max.) Burst Size:** Max. number of packets sent consecutively, ie over a very short period of time

**Idea:** packets sent by consuming tokens generated at constant rate  $r$

- limit input's
  - **Burst Size ( $b$ :** bucket capacity)
  - **Average Rate** (max admitted #packets over time period  $t$  is  $b+rt$ ).



aka: **Rate-Limiting methods**; can indicate which pkts to mark for ECN (Explicit Congestion Control, if NW protocol supports it)  
used for other problems too, eg prevent DoS attacks (e.g. Spotify apps to limit request-rates)

# Examples rate-limiting methods: the effect of buckets

input

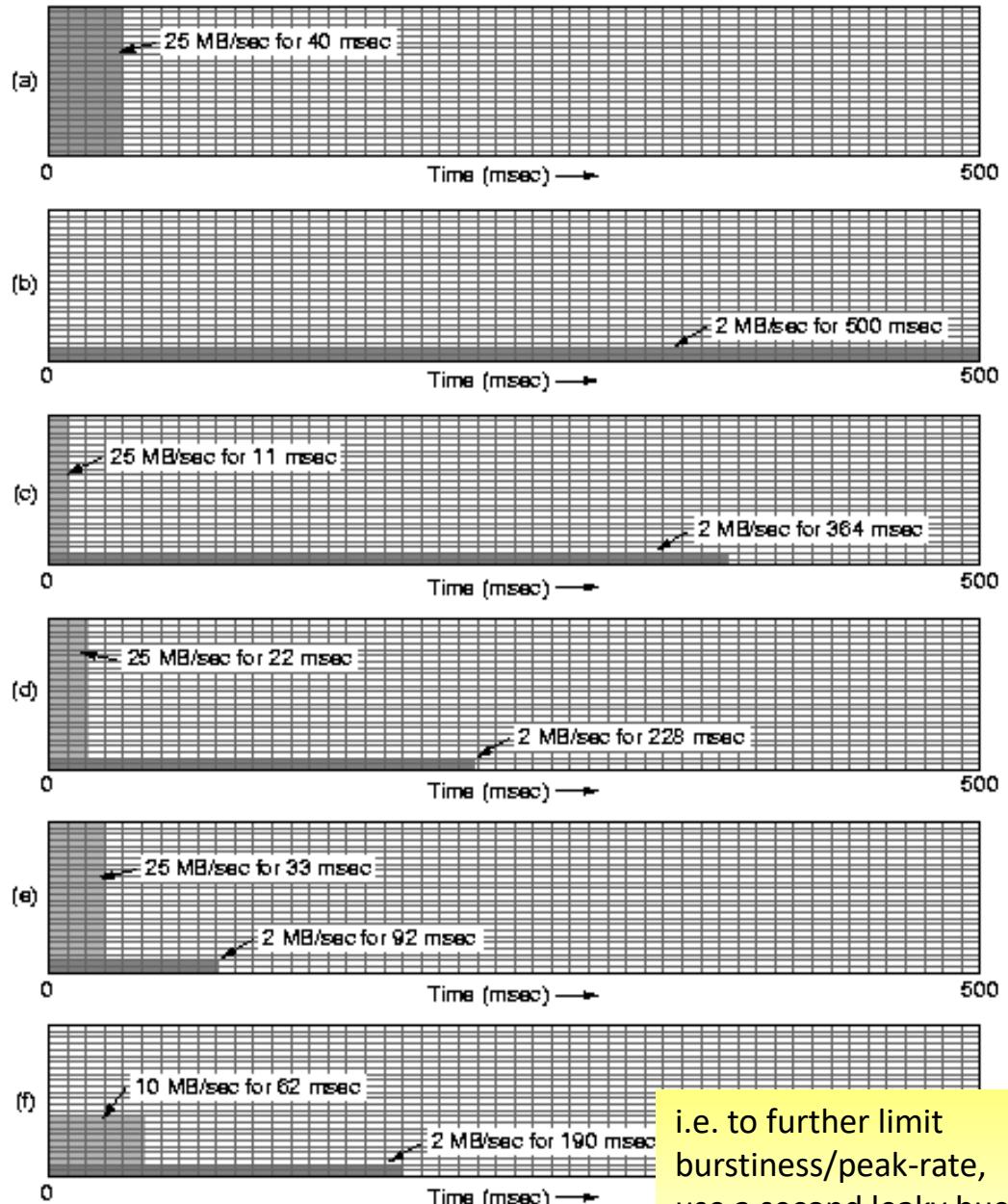
output 0KB token leaky bucket, 2MBps

output 250KB token leaky bucket, 2MBps

output 500KB token leaky bucket, 2MBps

output 750KB token leaky bucket, 2MBps

output token leaky bucket 500KB, 2MBps,  
feeding 0KB, 10MBps token leaky bucket



i.e. to further limit  
burstiness/peak-rate,  
use a second leaky bucket  
with higher rate

# A & B. Sidebar: Network Neutrality

---

What is network neutrality?

- *technical*: how an ISP should allocate/share its resources
  - packet scheduling, buffer management are the *mechanisms*
- *social, economic* principles
  - protecting free speech
  - encouraging innovation, competition
- enforced *legal* rules and policies

*Different countries have different approaches*

# Roadmap



Zoom in (again) inside a router: [Ch. 4.2.3-4.2.5]

- **Decisions that influence timing/bandwidth guarantees** (also related with congestion-control):  
Buffer management, packet scheduling, traffic shaping

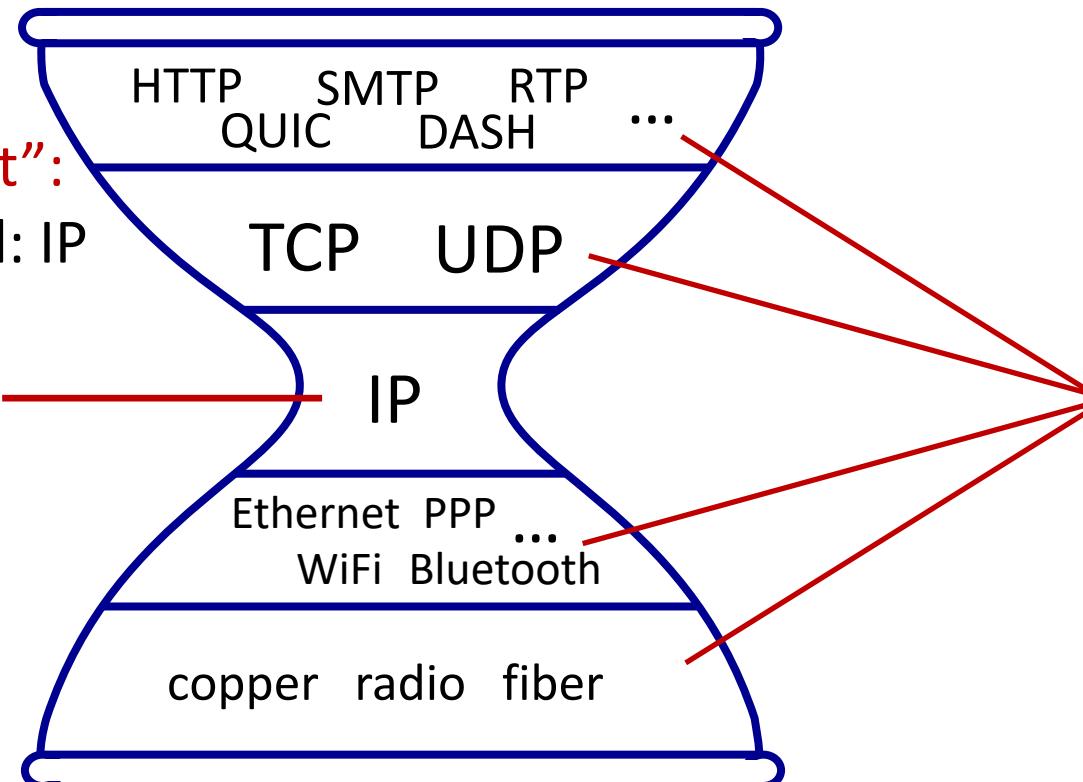
## ● Dealing with limitations of destination-based forwarding

- Software-Defined Networking (SDN) / NW Function Virtualization (NFV)[ch 4.4, 5.5, 4.5]
  - Concept
  - Generalized Forwarding- OpenFlow
  - Ctrl Plane – OpenFlow
- Middleboxes, "Intelligence" and controls
- Internet-of-Things in evolution: more types of traffic/devices...

# Recall: The IP hourglass

“Classic” Internet’s “thin waist”:

- one network layer protocol: IP
- must be implemented by every (billions) of Internet-connected devices
- Hence SIMPLE:
  - Prefix-matched destination-based forwarding



Read also: On the hourglass model, CACM 2019  
[https://www.youtube.com/watch?v=L9s096\\_r\\_U](https://www.youtube.com/watch?v=L9s096_r_U)

*many* protocols in physical, link, transport, and application layers

DOI:10.1145/3274770  
Used in the design of the Internet and Unix, the layered services of the hourglass model have enabled viral adoption and deployment scalability.

BY MICAH BECK

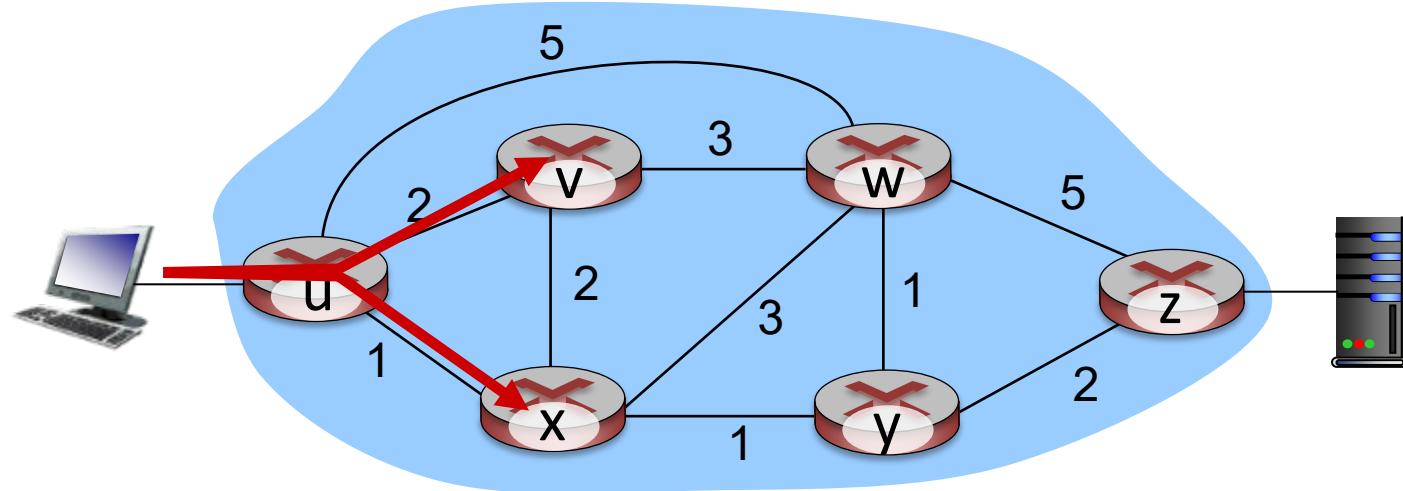
**On The Hourglass Model**

THE HOURGLASS MODEL of layered systems architecture is a visual and conceptual representation of an approach to design that seeks to support a great

layer by it tailored diff technologies in above an tooc as th net (see F ning layer of the Internet and DHCP)

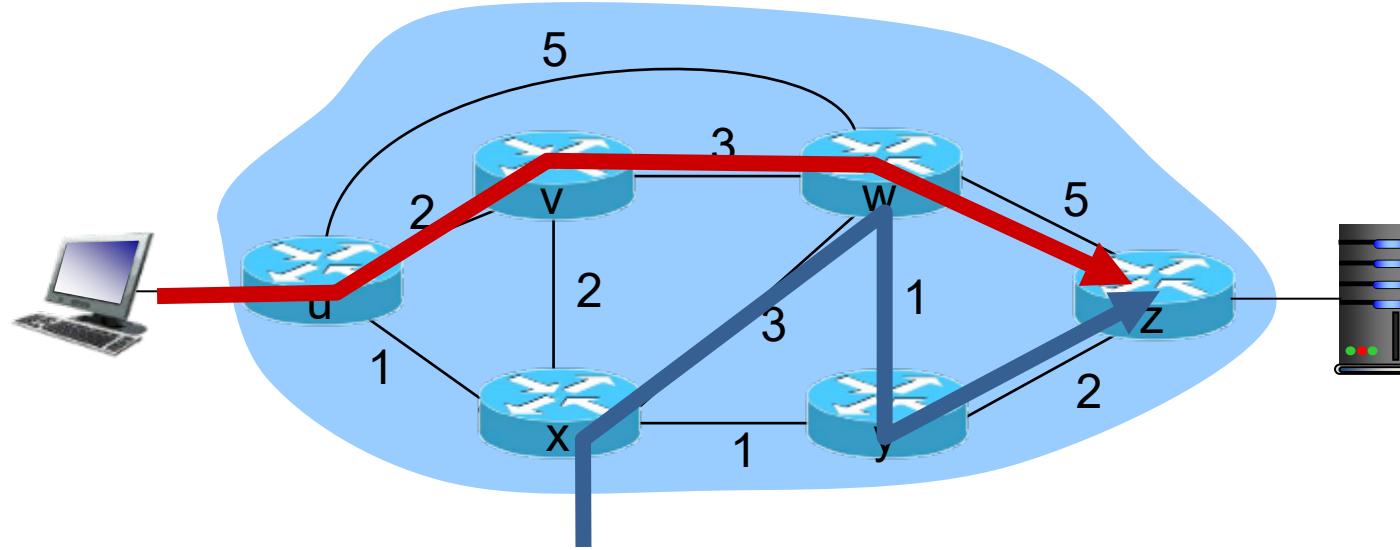
The sh glass mo the span various a mtable poros h glass is the intu functions is instr goals. Th are an hour waist of the restri large upp ing the i and supp The ho

# Difficulties with destination-based forwarding: (1)



Q: Can a network operator split u-to-z traffic and maintain 2 routes, along uvwz *and* uxyz (e.g. for load balancing and/or bandwidth guarantees)?  
A: can't do it (with destination-based forwarding)

# Difficulties with destination-based forwarding (2)



Q: what if w wants to route/forward blue and red traffic differently?

A: can't do it (with destination-based forwarding)

# Roadmap



Zoom in (again) inside a router: [Ch. 4.2.3-4.2.5]

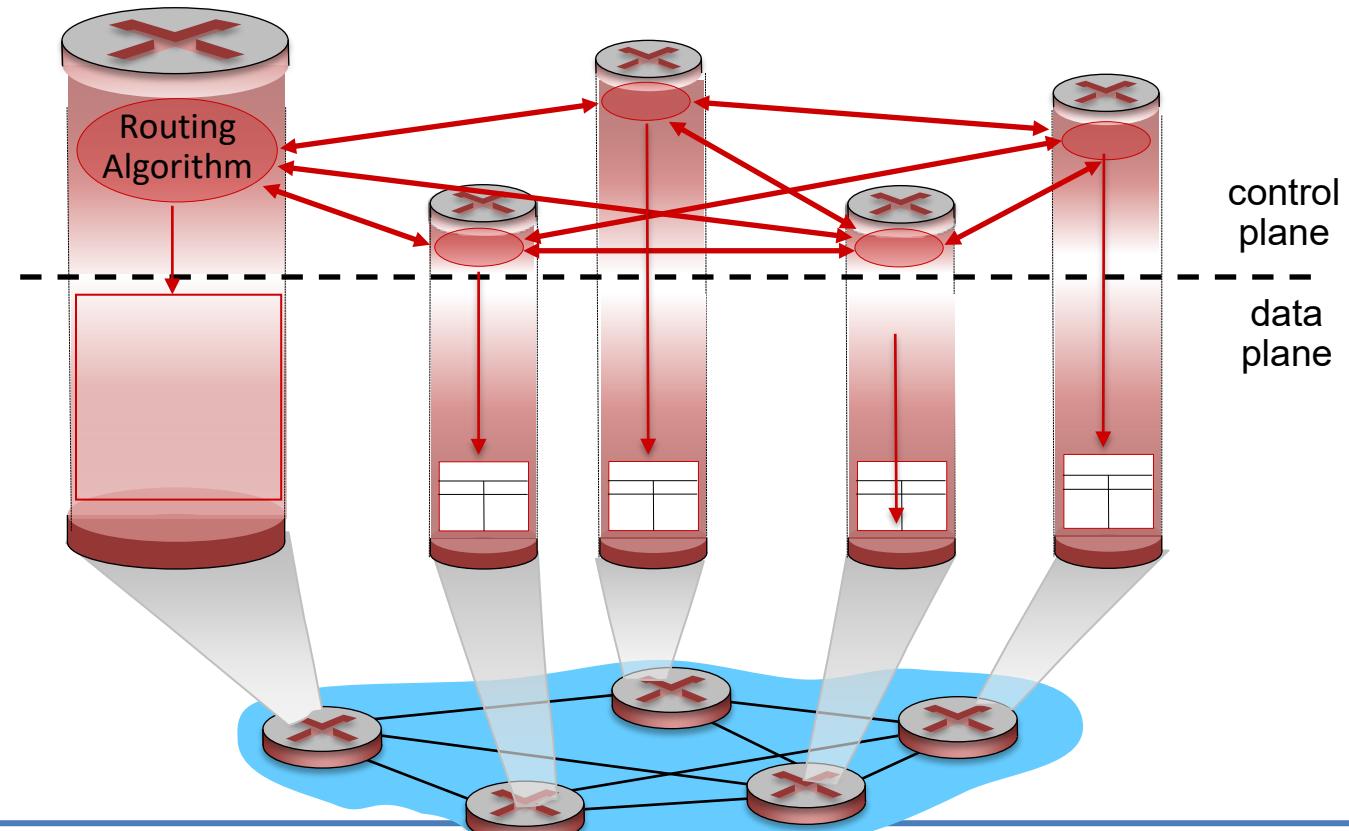
- **Decisions that influence timing/bandwidth guarantees** (also related with congestion-control):  
Buffer management, packet scheduling, traffic shaping

Dealing with limitations of destination-based forwarding

- **Software-Defined Networking (SDN) / NW Function Virtualization (NFV)**[ch 4.4, 5.5, 4.5]
  - Concept
  - Generalized Forwarding- OpenFlow
  - Ctrl Plane – OpenFlow
- Middleboxes, "Intelligence" and controls
- Internet-of-Things in evolution: more types of traffic/devices...

# Recall: Traditional Internet, per-router control plane

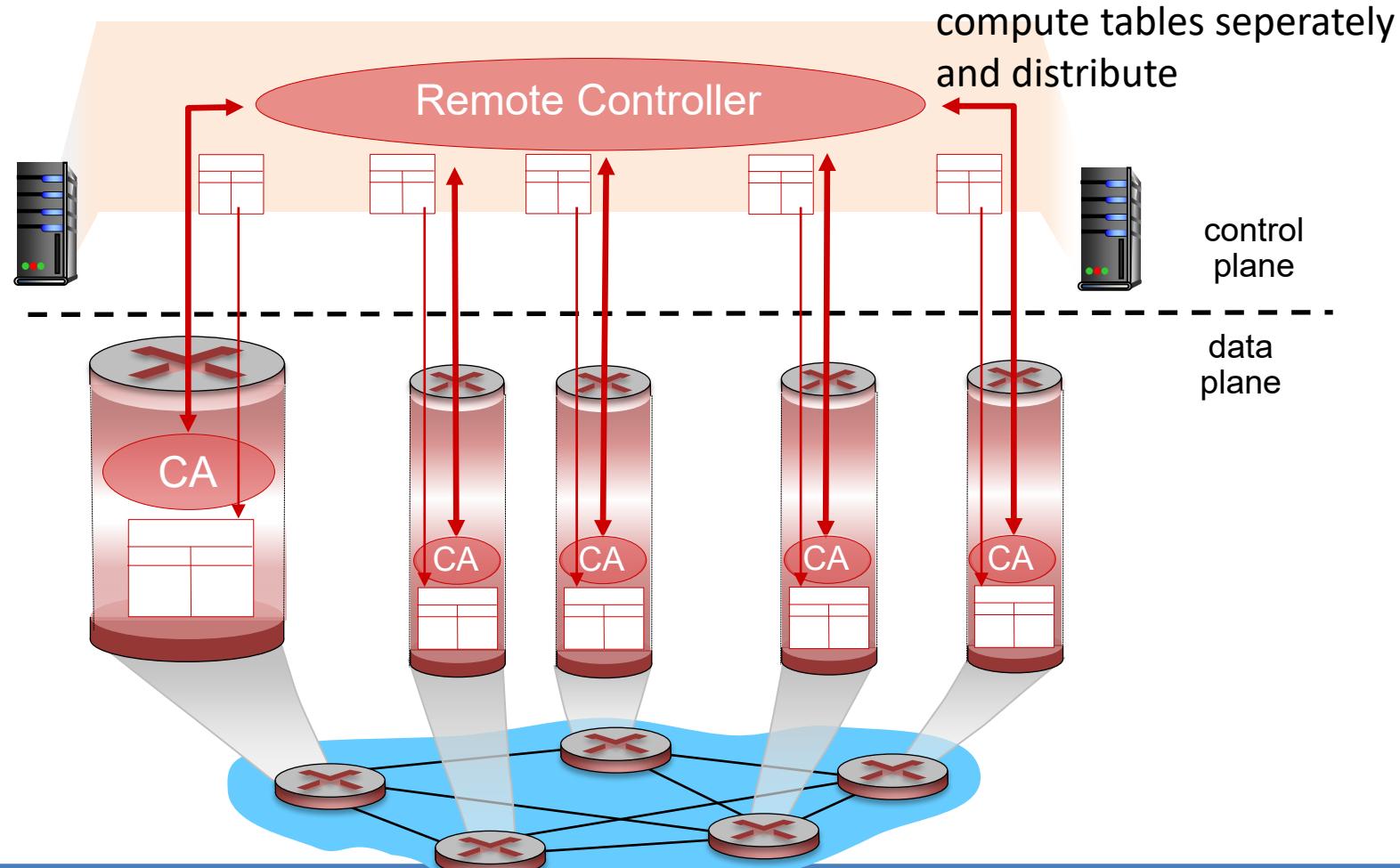
Individual routing algorithm components *in each and every router* interact with each other in control plane to compute forwarding tables



# Software defined networking (SDN)

## logically separated control plane

A distinct (typically remote) controller interacts with local **control agents** (CAs) in routers to compute forwarding tables



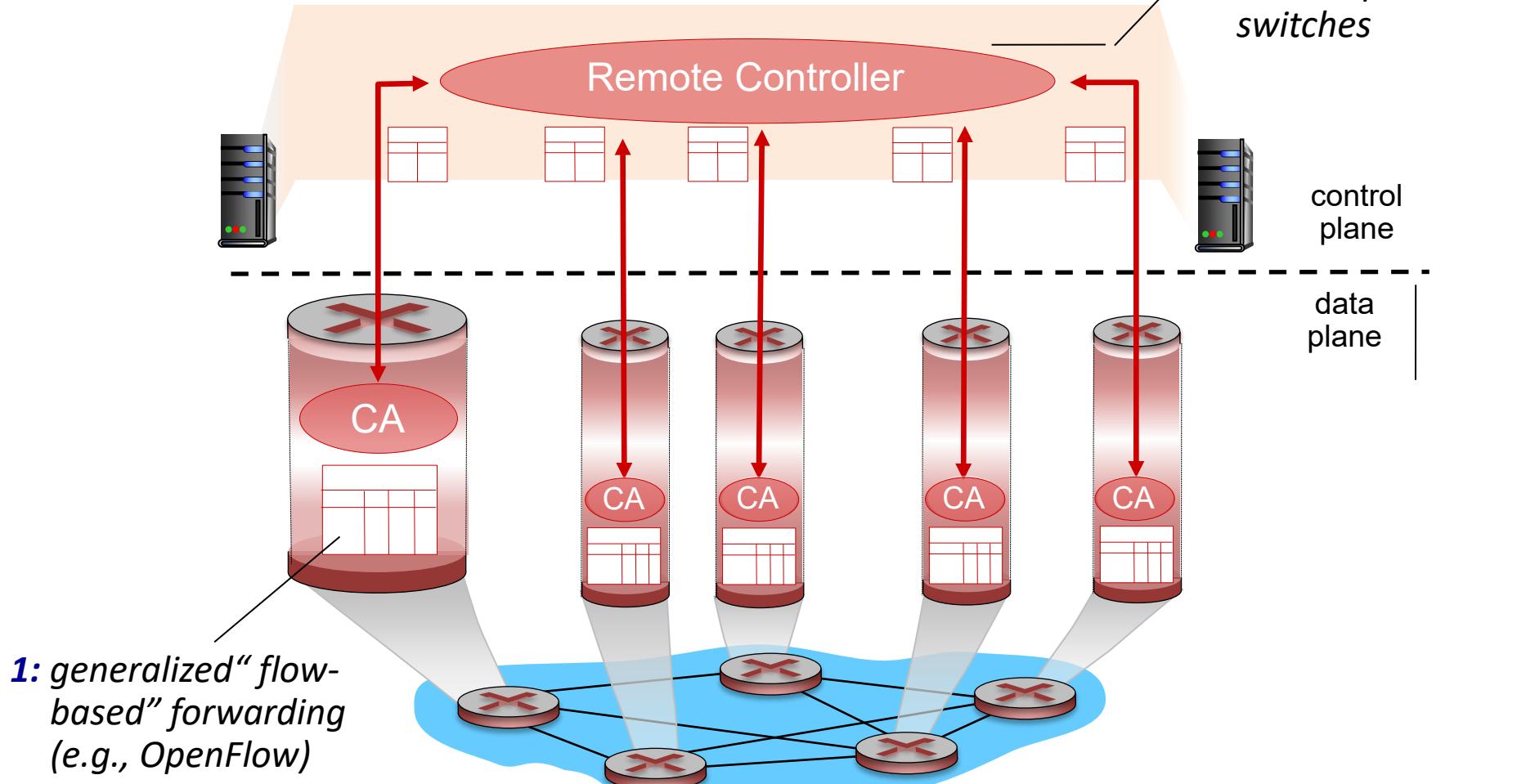
# Software defined networking (SDN)

associated term: Network Function Virtualization

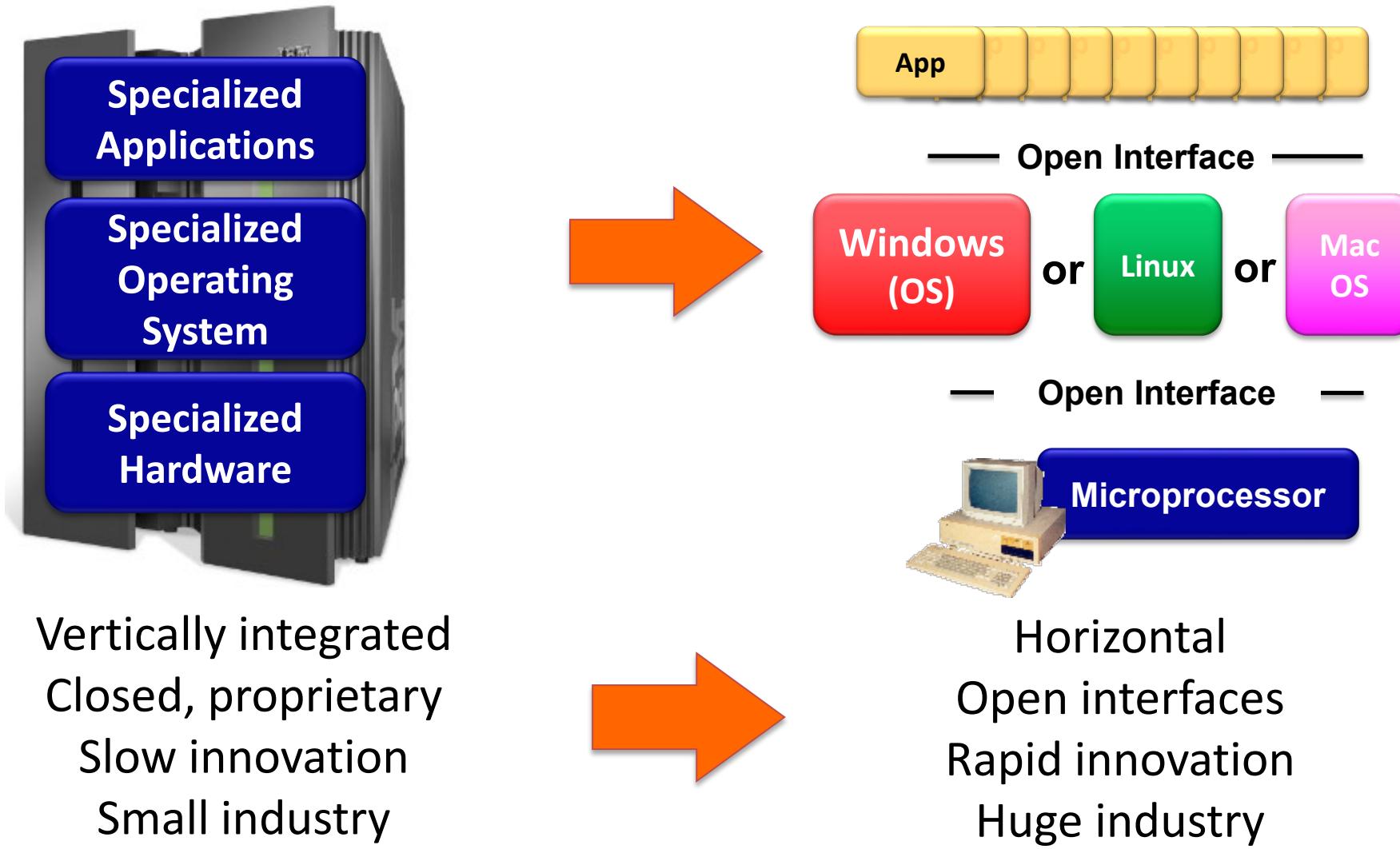
3. programmable  
control applications

routing  
access control  
...  
load balance

2. control plane  
functions external  
to data-plane  
switches



# Analogy: mainframe to PC evolution\*

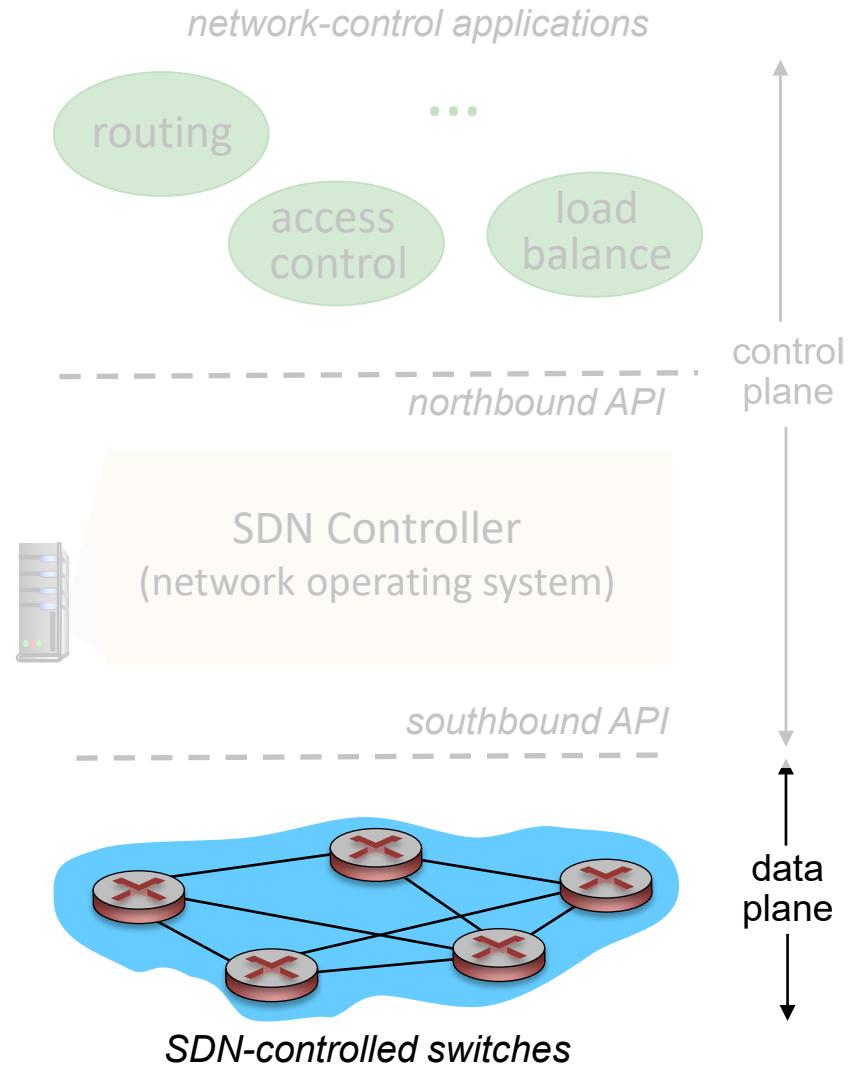


\* Slide courtesy: N. McKeown

# SDN perspective: data plane switches

## Data plane switches

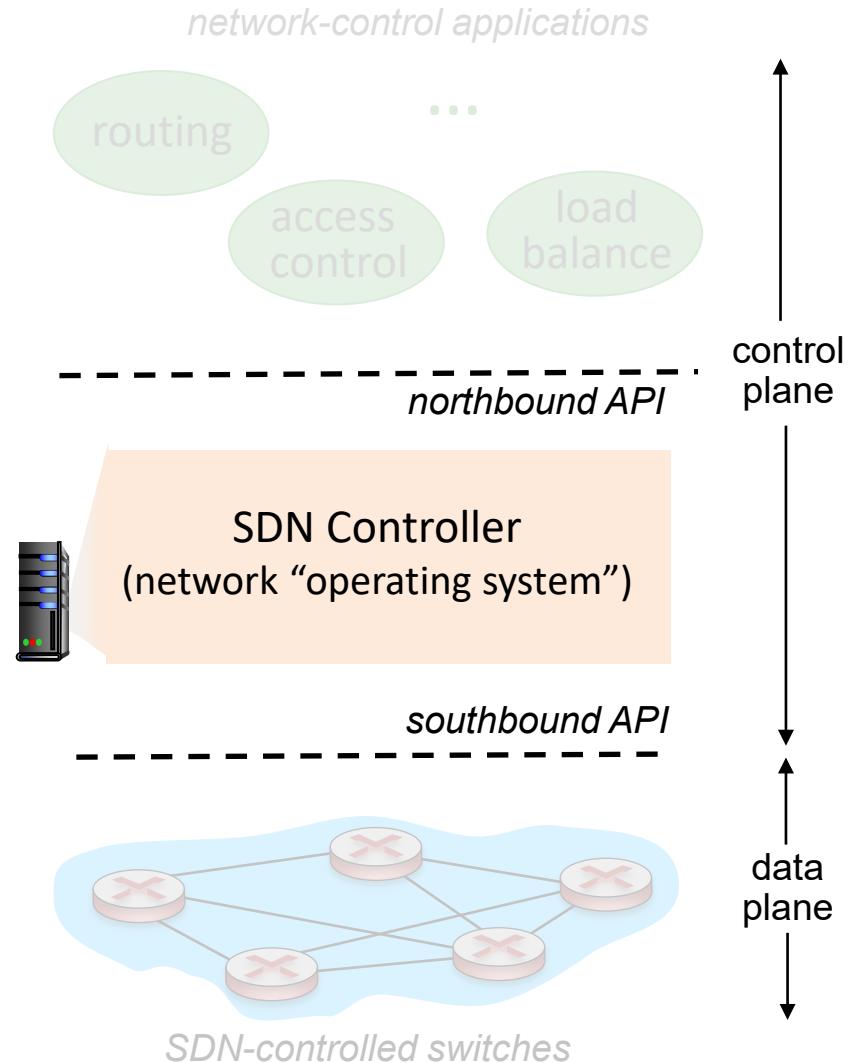
- fast, simple, for data-plane forwarding in H/W
- switch **flow table**: computed by controller
- API for table-based switch control (e.g., OpenFlow)
- protocol for communicating with controller (e.g., OpenFlow)



# SDN perspective: SDN controller

## SDN controller (*network OS*):

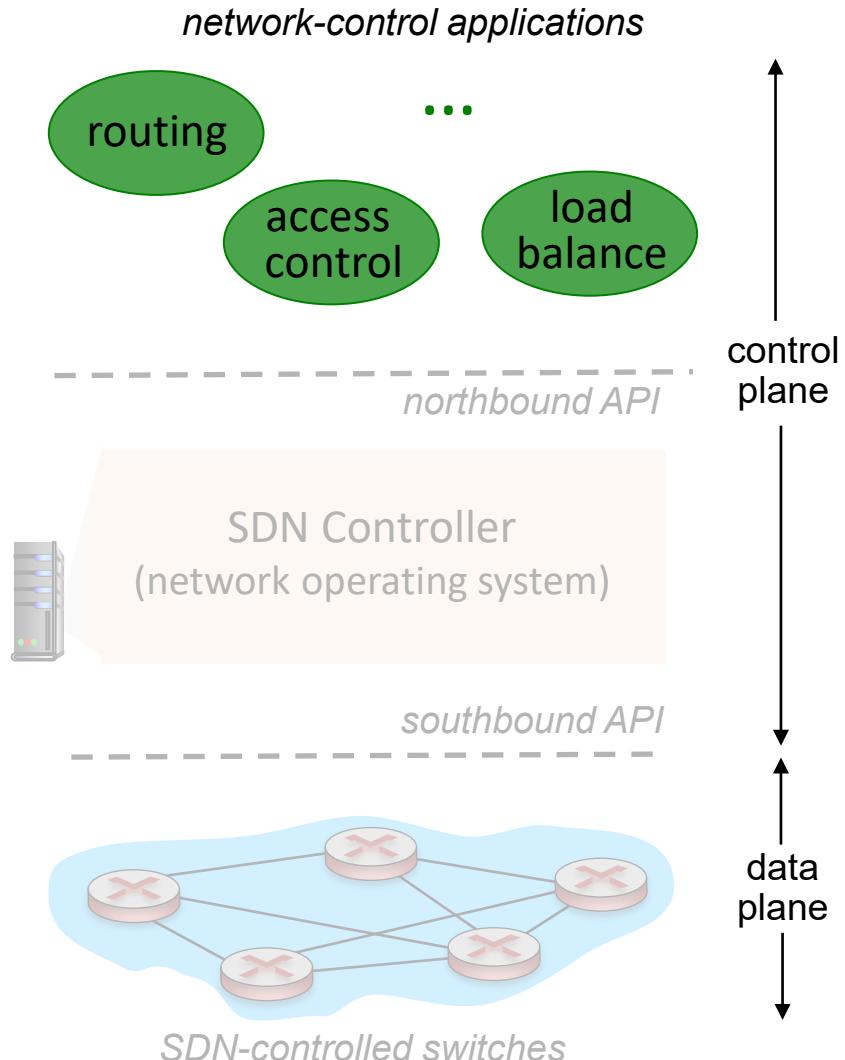
- maintains network state information
- interacts with network control applications “above” (northbound API)
- interacts with network switches “below” (southbound API)
- implemented as ***distributed system*** for performance, scalability, robustness



# SDN perspective: control applications

## *network-control apps:*

- “brains” of control: for control functions using lower-level services, API provided by SDN controller
- can be provided by 3<sup>rd</sup> party: distinct from routing vendor, or SDN controller



# Roadmap



Zoom in (again) inside a router: [Ch. 4.2.3-4.2.5]

- **Decisions that influence timing/bandwidth guarantees** (also related with congestion-control):  
Buffer management, packet scheduling, traffic shaping

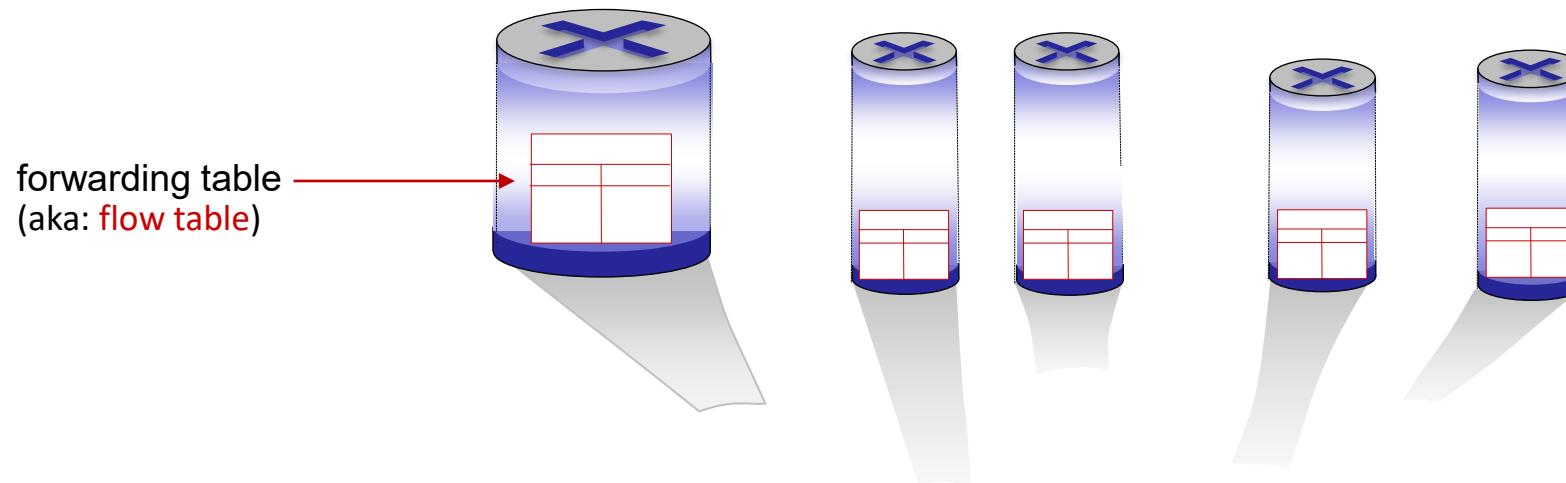
Dealing with limitations of destination-based forwarding

- **Software-Defined Networking (SDN) / NW Function Virtualization (NFV)**[ch 4.4, 5.5, 4.5]
  - Concept
  - Generalized Forwarding- OpenFlow
  - Ctrl Plane – OpenFlow
- Middleboxes, "Intelligence" and controls
- Internet-of-Things in evolution: more types of traffic/devices...

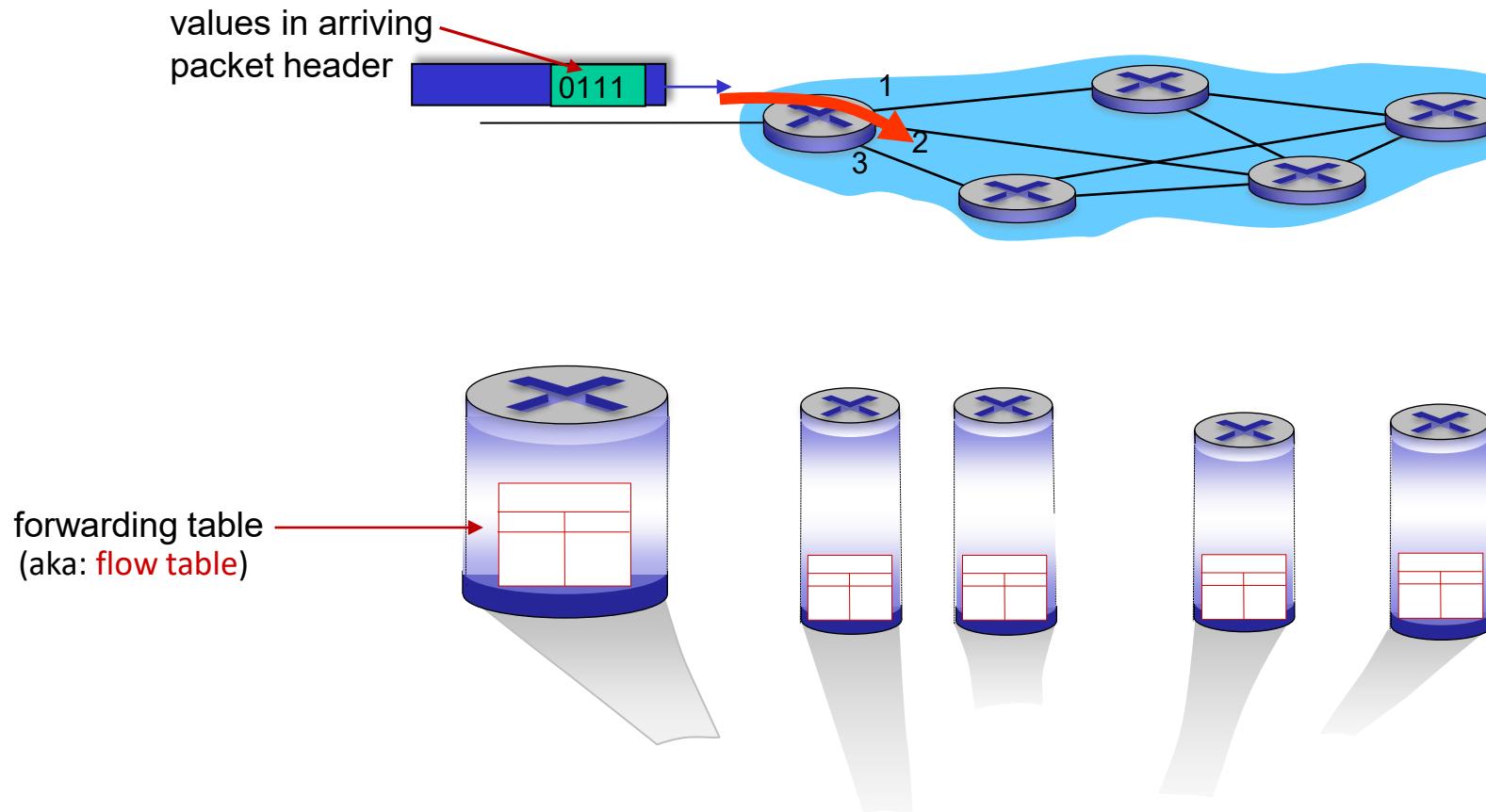
# Generalized forwarding: match plus action

*Recall: traditional router contains a **forwarding table** (aka: **flow table**)*

- “match plus action” abstraction: match bits in arriving packet, take action
  - *destination-based forwarding*: forward based on dest. IP address
  - *generalized forwarding*:
    - many header fields can determine action
    - many actions possible: drop/copy/modify/log packet

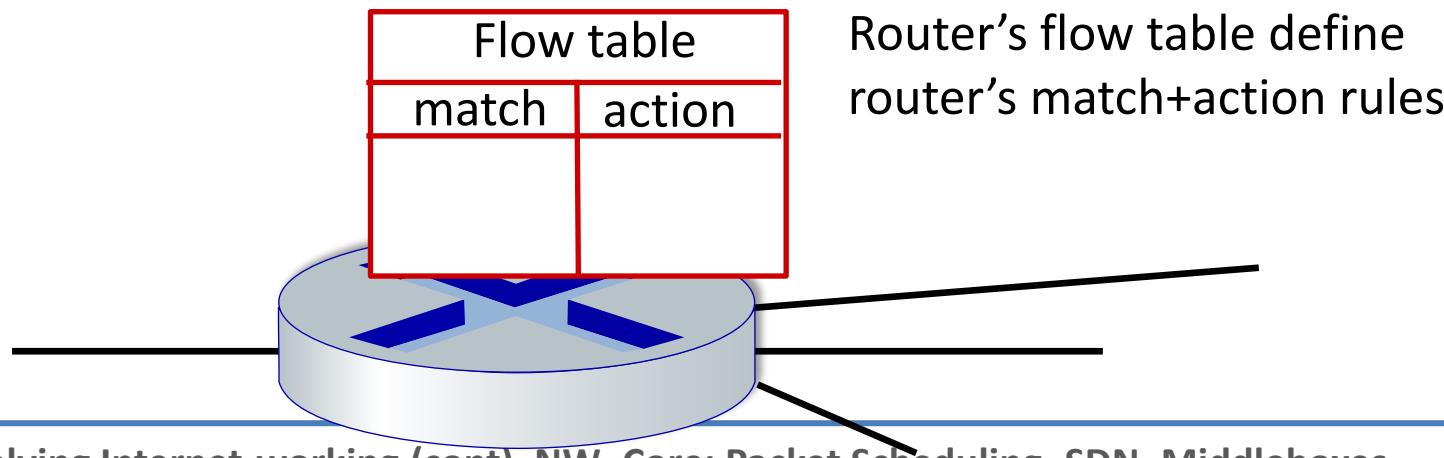


# Generalized forwarding: match plus action



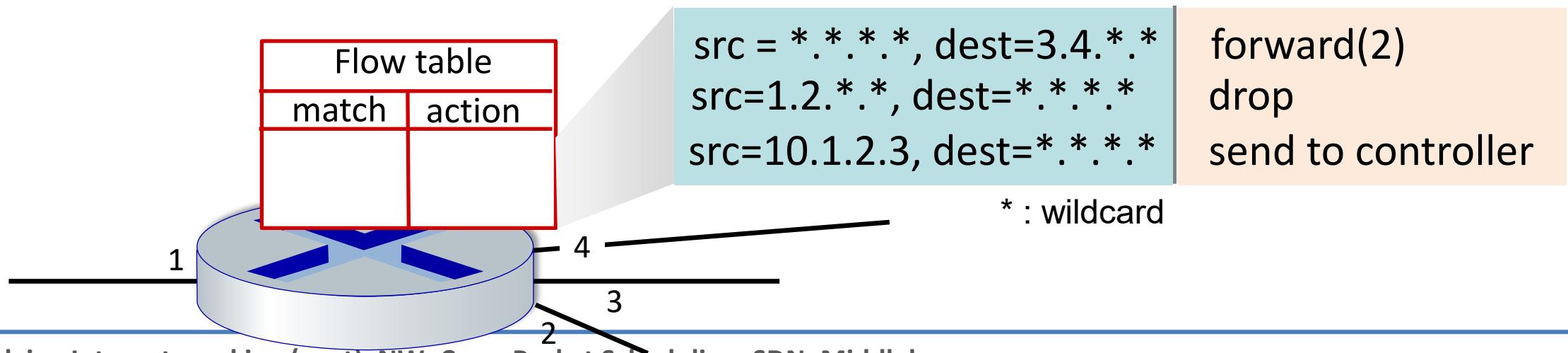
# Flow table abstraction

- **flow:** defined by header field values (in link-, network-, transport-layer fields)
- **generalized forwarding:** simple packet-handling rules
  - **match:** pattern values in packet header fields
  - **actions:** for matched packet: drop, forward, modify or send matched packet to controller

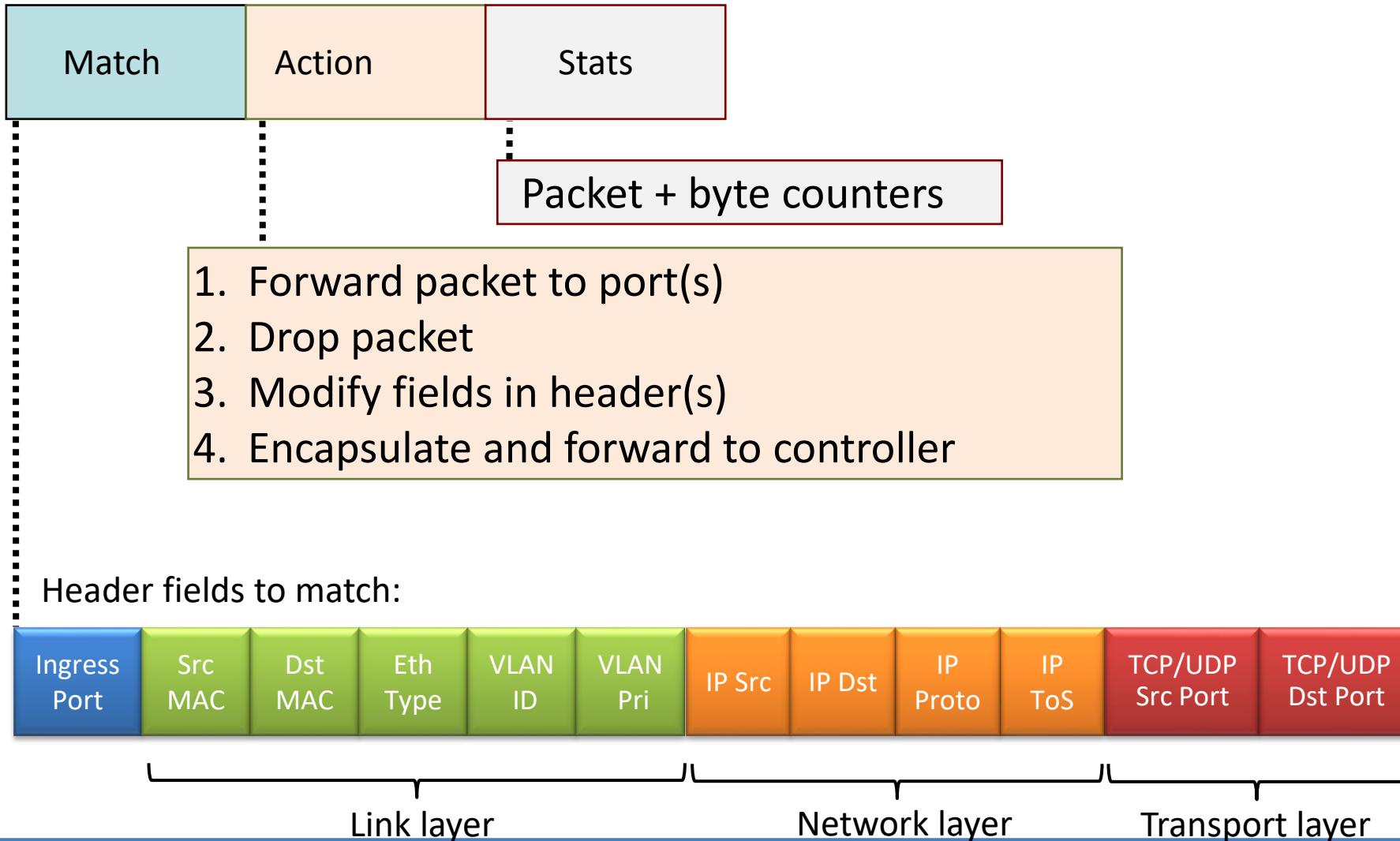


# Flow table abstraction

- **flow:** defined by header fields
- **generalized forwarding:** simple packet-handling rules
  - **match:** pattern values in packet header fields
  - **actions:** for matched packet: drop, forward, modify, or send matched packet to controller



# OpenFlow: flow table entries



# OpenFlow: examples

## Destination-based forwarding:

| Switch Port | MAC src | MAC dst | Eth type | VLAN ID | VLAN Pri | IP Src | IP Dst   | IP Prot | IP ToS | TCP s-port | TCP d-port | Action |
|-------------|---------|---------|----------|---------|----------|--------|----------|---------|--------|------------|------------|--------|
| *           | *       | *       | *        | *       | *        | *      | 51.6.0.8 | *       | *      | *          | *          | port6  |

IP datagrams destined to IP address 51.6.0.8 should be forwarded to router output port 6

## Firewall:

| Switch Port | MAC src | MAC dst | Eth type | VLAN ID | VLAN Pri | IP Src | IP Dst | IP Prot | IP ToS | TCP s-port | TCP d-port | Action |
|-------------|---------|---------|----------|---------|----------|--------|--------|---------|--------|------------|------------|--------|
| *           | *       | *       | *        | *       | *        | *      | *      | *       | *      | *          | 22         | drop   |

Block (do not forward) all datagrams destined to TCP port 22 (ssh port #)

| Switch Port | MAC src | MAC dst | Eth type | VLAN ID | VLAN Pri | IP Src      | IP Dst | IP Prot | IP ToS | TCP/UDP s-port | TCP/UDP d-port | Action |
|-------------|---------|---------|----------|---------|----------|-------------|--------|---------|--------|----------------|----------------|--------|
| *           | *       | *       | *        | *       | *        | 128.119.1.1 | *      | *       | *      | *              | *              | drop   |

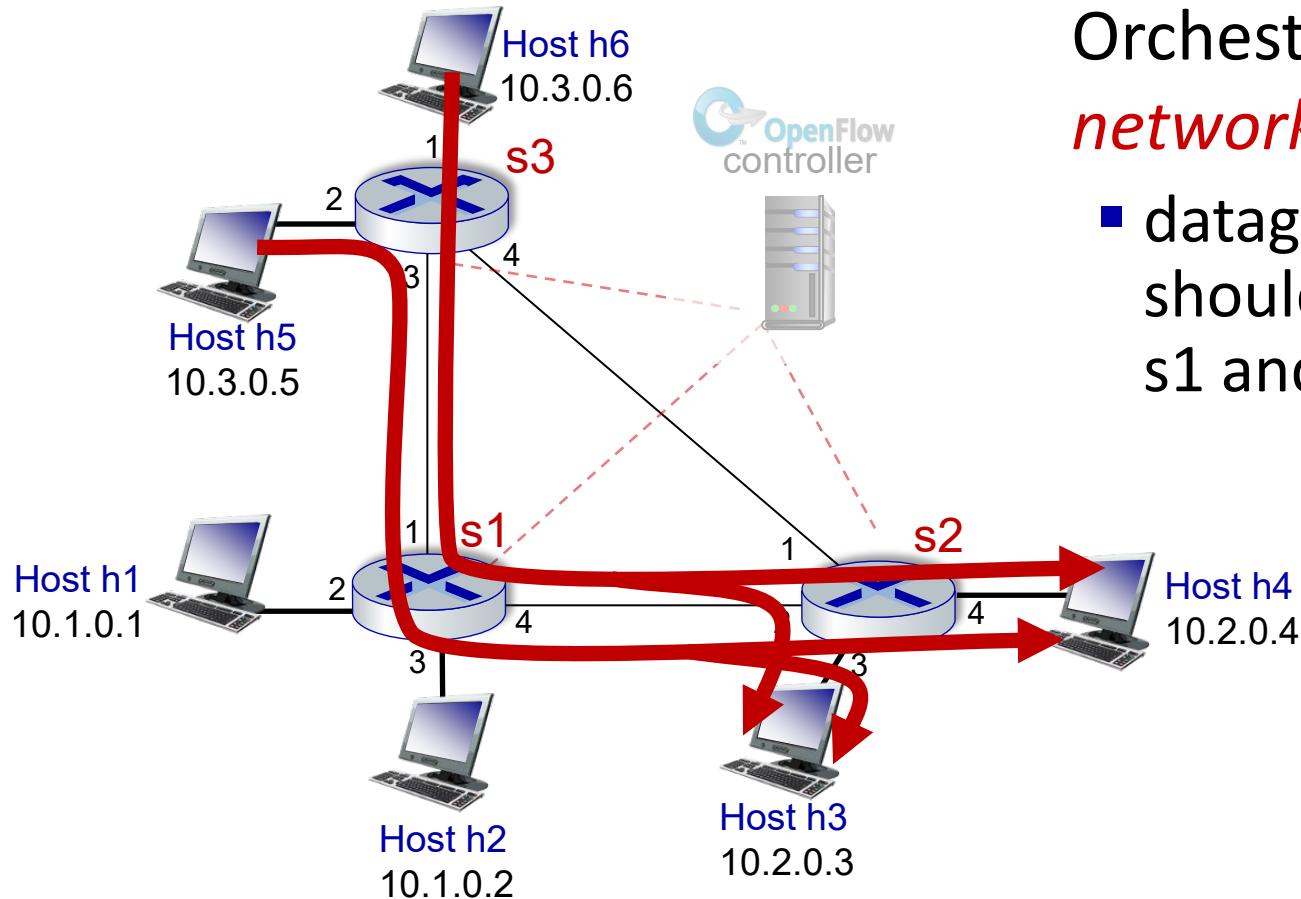
Block (do not forward) all datagrams sent by host 128.119.1.1

## Layer 2 destination-based forwarding:

| Switch Port | MAC src | MAC dst               | Eth type | VLAN ID | VLAN Pri | IP Src | IP Dst | IP Prot | IP ToS | TCP s-port | TCP d-port | Action |
|-------------|---------|-----------------------|----------|---------|----------|--------|--------|---------|--------|------------|------------|--------|
| *           | *       | 22:A7:23:<br>11:E1:02 | *        | *       | *        | *      | *      | *       | *      | *          | *          | port3  |

layer 2 frames with destination MAC address 22:A7:23:11:E1:02 should be forwarded to output port 3

# OpenFlow example

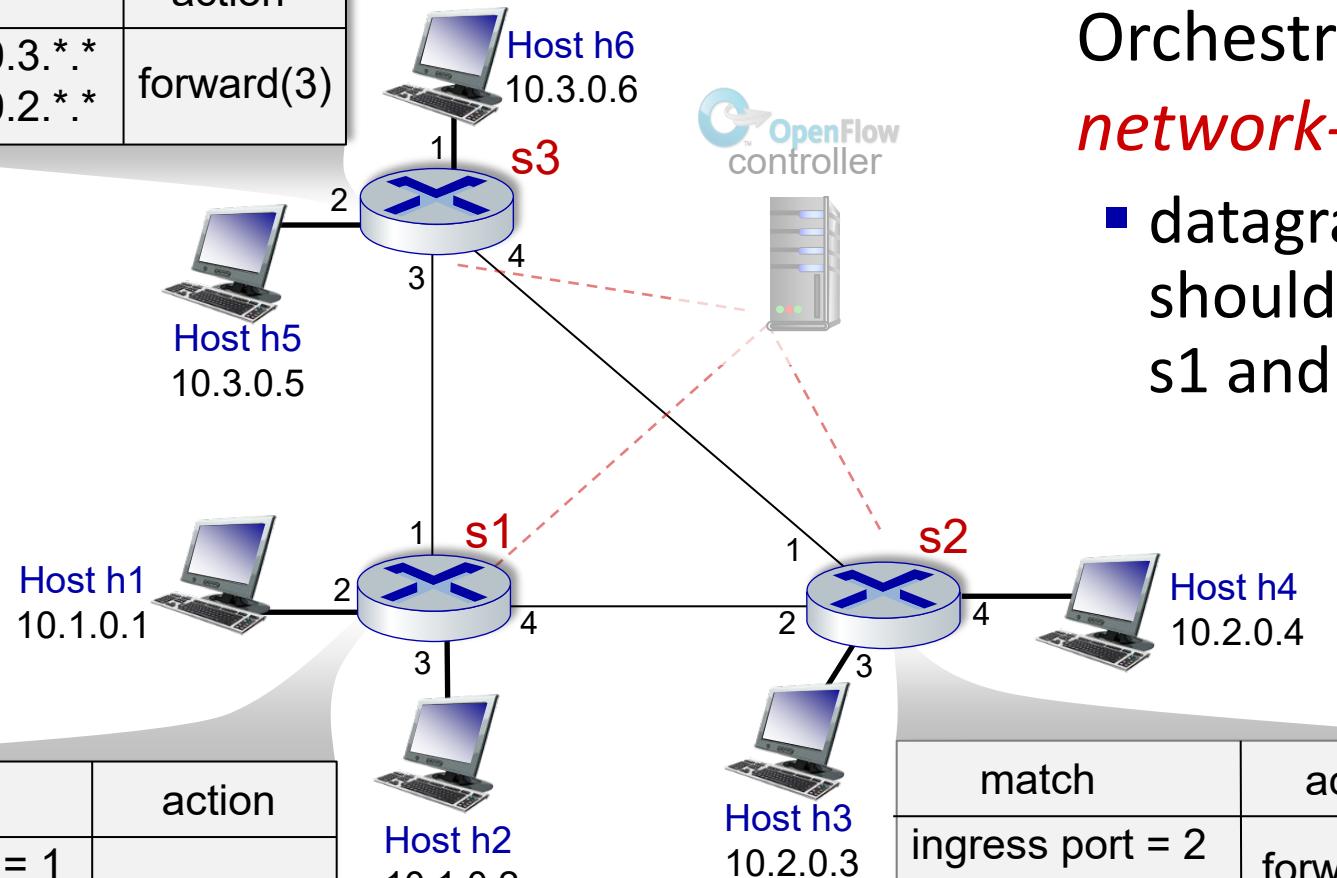


Orchestrated tables can create *network-wide* behavior, e.g.,:

- datagrams from hosts h5, h6 should be sent to h3 or h4, via s1 and from there to s2

# OpenFlow example

| match             | action     |
|-------------------|------------|
| IP Src = 10.3.*.* |            |
| IP Dst = 10.2.*.* | forward(3) |



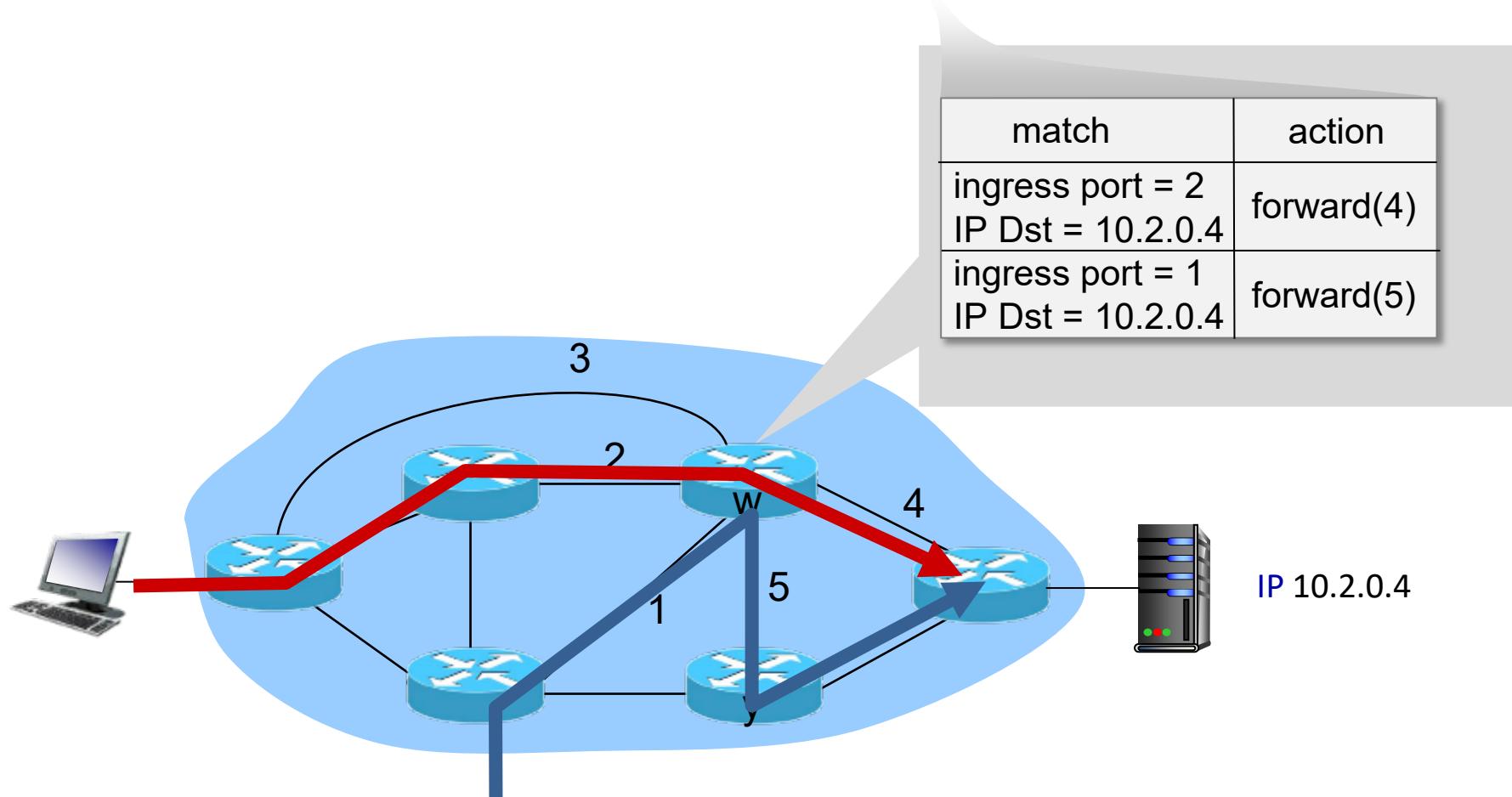
| match             | action     |
|-------------------|------------|
| ingress port = 1  |            |
| IP Src = 10.3.*.* | forward(4) |
| IP Dst = 10.2.*.* |            |

Orchestrated tables can create *network-wide* behavior, e.g.,:

- datagrams from hosts h5, h6 should be sent to h3 or h4, via s1 and from there to s2

| match             | action     |
|-------------------|------------|
| ingress port = 2  |            |
| IP Dst = 10.2.0.3 | forward(3) |
| ingress port = 2  |            |
| IP Dst = 10.2.0.4 | forward(4) |

# With generalized forwarding ...



w can forward blue and red traffic differently!

# Roadmap



Zoom in (again) inside a router: [Ch. 4.2.3-4.2.5]

- **Decisions that influence timing/bandwidth guarantees** (also related with congestion-control):  
Buffer management, packet scheduling, traffic shaping

**Dealing with limitations of destination-based forwarding**

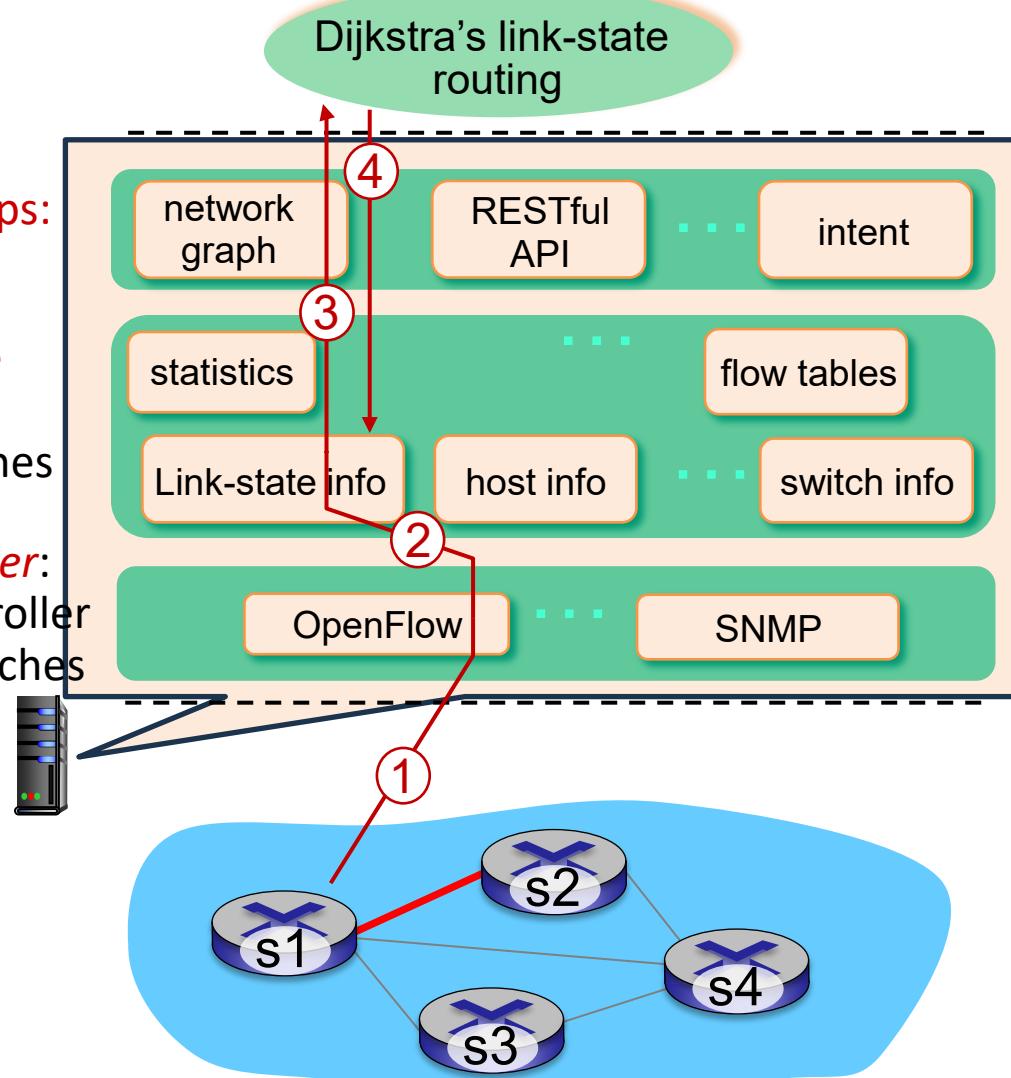
- **Traffic Engineering & MPLS [Ch. 6.5.1]**
- **Software-Defined Networking (SDN) / NW Function Virtualization (NFV)[ch 4.4, 5.5, 4.5]**
  - Concept
  - Generalized Forwarding- OpenFlow
  - Ctrl Plane – OpenFlow
- Middleboxes, "Intelligence" and controls
- Internet-of-Things in evolution: more types of traffic/devices...

# SDN controller: control/data plane interaction example

Interface layer to network control apps: abstractions API

Network-wide state management layer: state of links, switches

*Communication layer:* between SDN controller and controlled switches



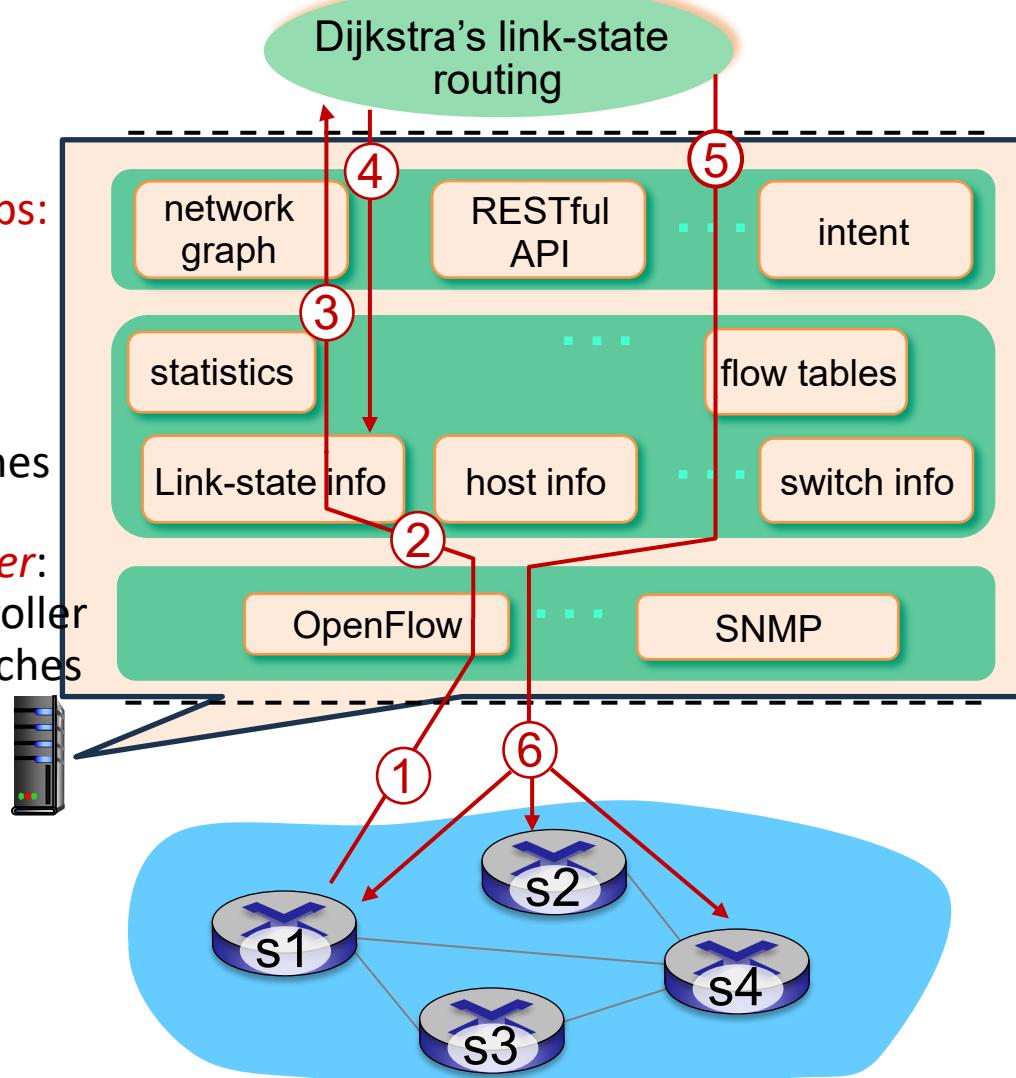
- ① S1, experiencing link failure uses *OpenFlow port status message* to notify controller
- ② SDN controller receives OpenFlow message, *updates link status info*
- ③ Dijkstra's routing algorithm application has previously registered to be *called whenever link status changes*. It is called.
- ④ Dijkstra's routing algorithm access network graph info, link state info in controller, *computes new routes*

# SDN: control/data plane interaction example

Interface layer to network control apps: abstractions API

Network-wide state management layer: state of links, switches

Communication layer: between SDN controller and controlled switches



⑤ link state routing app interacts with flow-table-computation component in SDN controller, which *computes new flow tables*

⑥ controller uses OpenFlow to *install new tables* in switches that need updating

# Roadmap



Zoom in (again) inside a router: [Ch. 4.2.3-4.2.5]

- **Decisions that influence timing/bandwidth guarantees** (also related with congestion-control):  
Buffer management, packet scheduling, traffic shaping

Dealing with limitations of destination-based forwarding

- **Software-Defined Networking (SDN) / NW Function Virtualization (NFV)**[ch 4.4, 5.5, 4.5]
  - Concept
  - Generalized Forwarding- OpenFlow
  - Ctrl Plane – OpenFlow
- **Middleboxes**, "Intelligence" and controls
- **Internet-of-Things in evolution: more types of traffic/devices...**

# Middleboxes

---

Middlebox (RFC 3234)

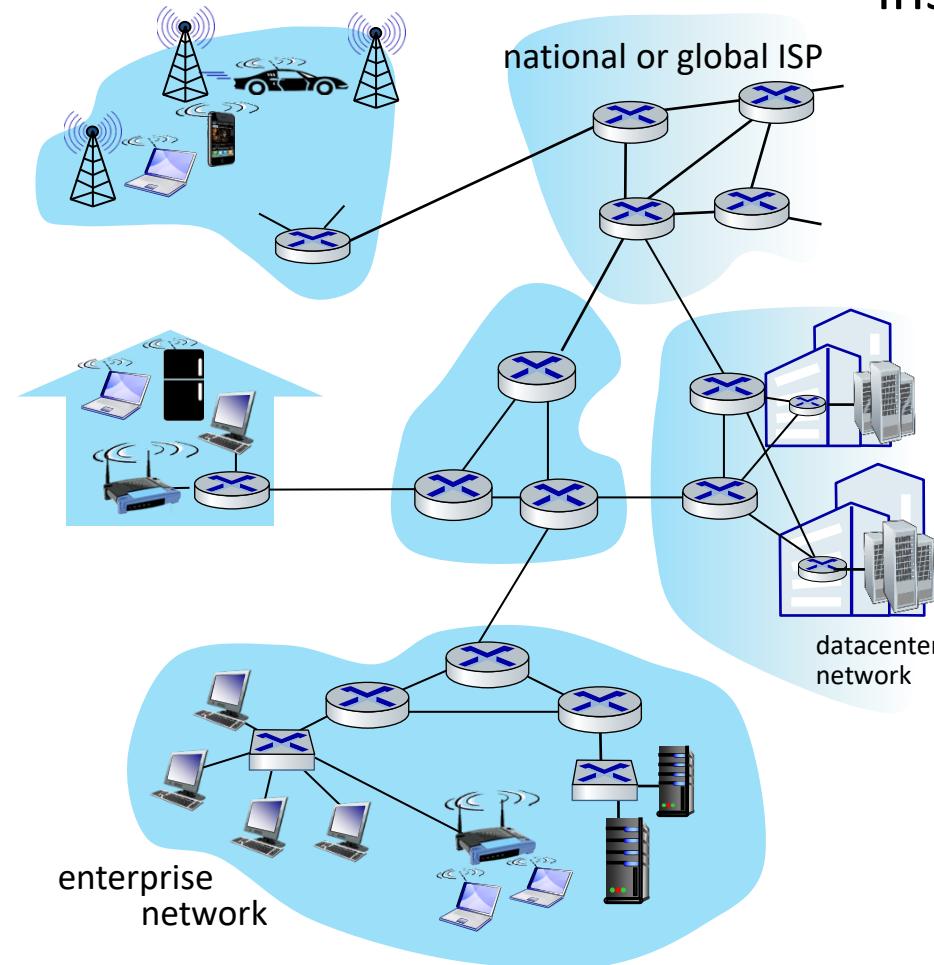
“any intermediary box performing functions apart from normal, standard functions of an IP router on the data path between a source host and destination host”

# Middleboxes everywhere!

initially: proprietary (closed) hardware solutions...

**NAT**: home, cellular, institutional

**Application-specific**: service providers, institutional, CDN



**Firewalls, IDS**: corporate, institutional, service providers, ISPs

**Load balancers**: corporate, service provider, data center, mobile nets

**Caches**: service provider, mobile, CDNs

# Note: OpenFlow abstraction

- **match+action:** abstraction unifies different kinds of devices

## “Classic” router

- *match:* longest destination IP prefix
- *action:* forward out a link

## Switch

- *match:* destination MAC address
- *action:* forward or flood

...move towards “whitebox” hardware implementing open API

## Firewall

- *match:* IP addresses and TCP/UDP port numbers
- *action:* permit or deny

## NAT

- *match:* IP address and port
- *action:* rewrite address and port

simple form of “network programmability”: programmable, per-packet “processing”  
P4 (see p4.org), programmable/smart NIC

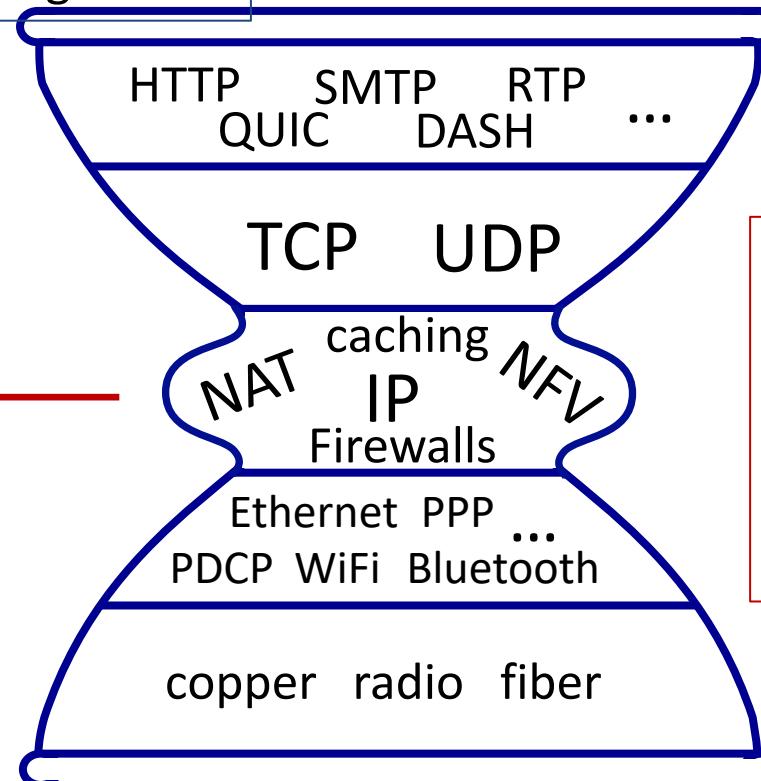
# The IP hourglass, at middle age

Cornerstones of early Internet:

- simple connectivity –IP: that narrow waist
- intelligence, complexity at NW edge

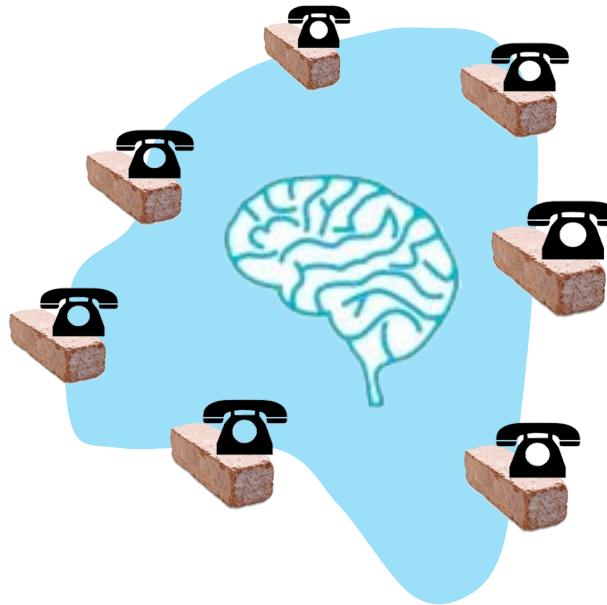
Internet's middle age  
“love handles”?

- middleboxes,  
operating inside the  
network



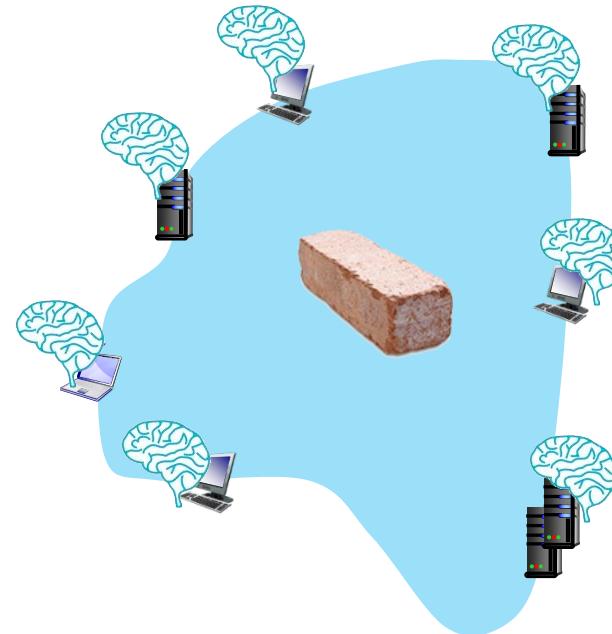
- SDN: (logically) centralized control and management often in private/public cloud
- network functions virtualization (NFV): programmable services over white box networking, computation, storage

# Where was/is/goes “intelligence”?



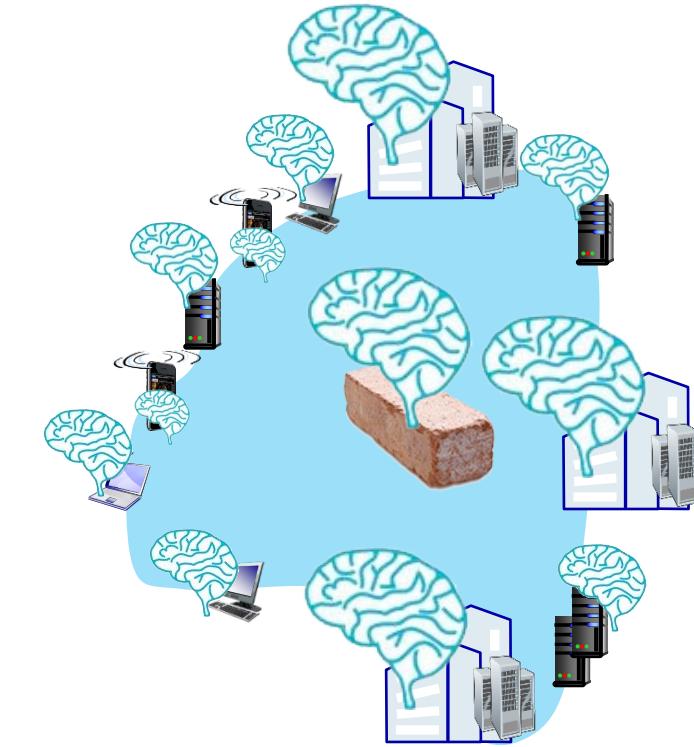
## 20<sup>th</sup> century phone net:

- intelligence/computing at network switches



## Internet (pre-2005)

- intelligence, computing at edge

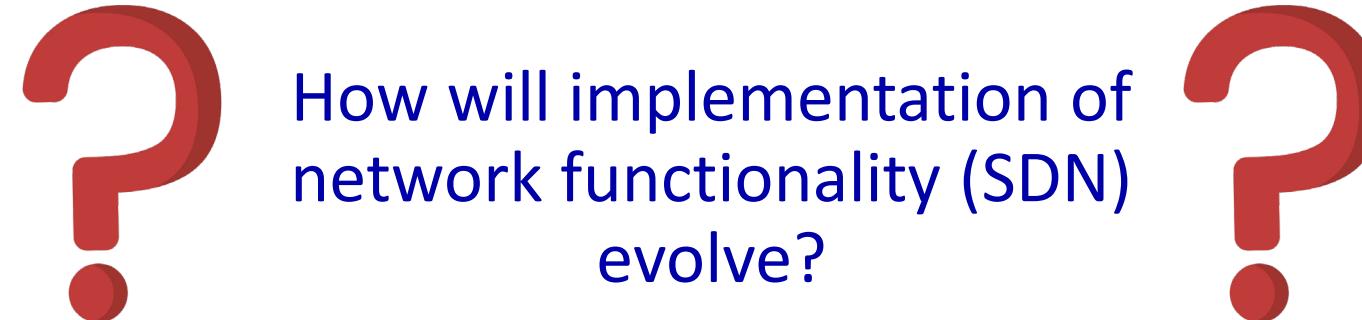


## Internet (post-2005)

- programmable network devices
- intelligence, computing, massive application-level infrastructure at edge

# SDN challenges and the future of traditional network protocols

- **hardening the control plane:** dependable, reliable, performance-scalable, secure distributed system: leverage strong theory of reliable distributed system for control plane
- networks, protocols meeting **mission-specific/critical** requirements
  - e.g., real-time, ultra-reliable, ultra-secure; critical in 5G (& 6G research for) cellular networks
- SDN-computed versus router-computed forwarding tables:
  - just one example of **logically-centralized-computed** versus distributed/peer-computed
- one could imagine **SDN-computed congestion control:**
  - controller sets sender rates based on router-reported (to controller) congestion levels



How will implementation of network functionality (SDN) evolve?

# Roadmap



Zoom in (again) inside a router: [Ch. 4.2.3-4.2.5]

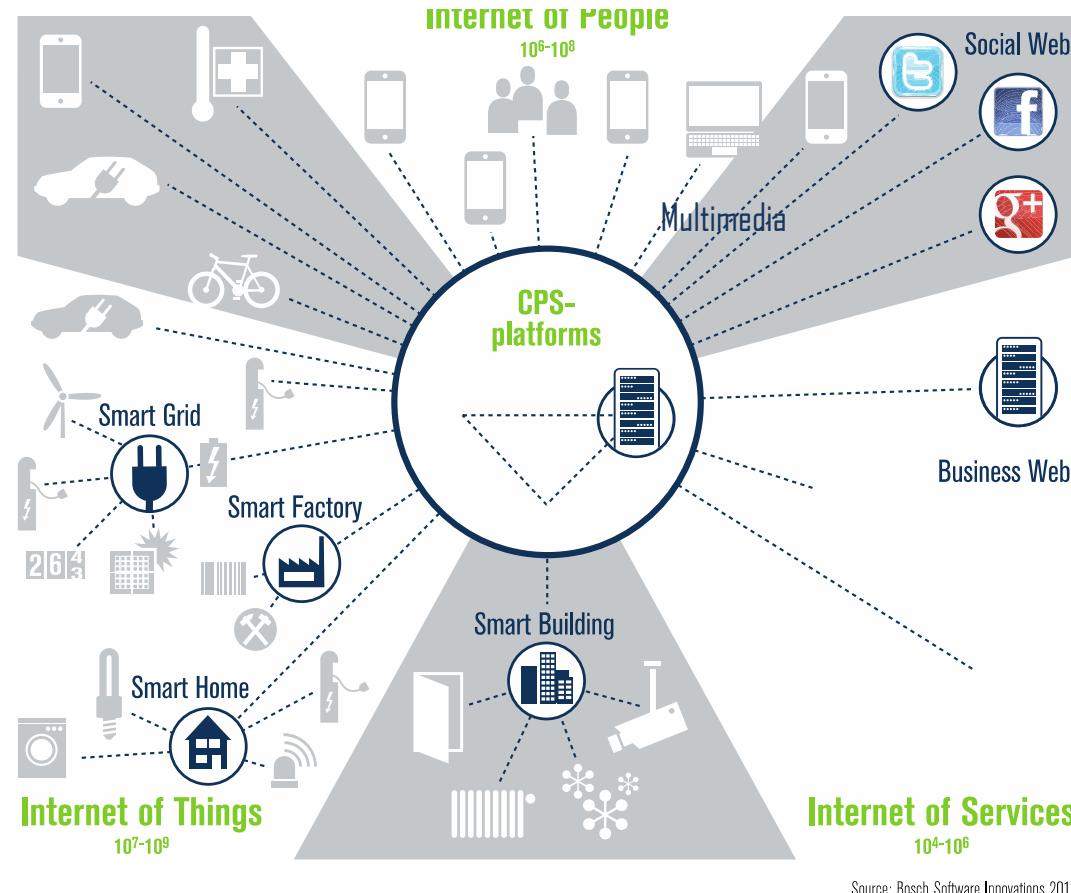
- **Decisions that influence timing/bandwidth guarantees** (also related with congestion-control):  
Buffer management, packet scheduling, traffic shaping

Dealing with limitations of destination-based forwarding

- **Software-Defined Networking (SDN) / NW Function Virtualization (NFV)**[ch 4.4, 5.5, 4.5]
  - Concept
  - Generalized Forwarding- OpenFlow
  - Ctrl Plane – OpenFlow
- **Middleboxes**, "Intelligence" and controls
- **Internet-of-Things in evolution: more types of traffic/devices...**



# IoT in evolution: more types of traffic/devices...



continuous evolution ....

# Further study – food for thought: Continuous evolution (continued)

e.g. Industrial IoT/EtherCAT, ..., "Smart" programmable NIC, NW compute fabric, ...

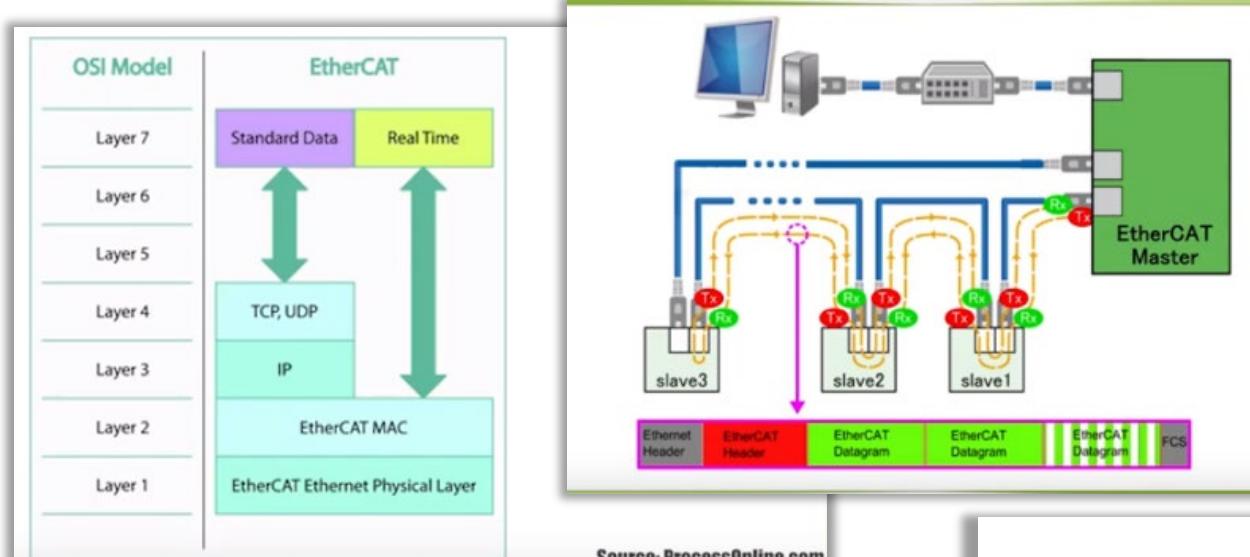
"New" protocols: Ethernet Control Automation Systems

<https://www.ethercat.org/en/technology.html>

<https://www.youtube.com/watch?v=i-BwnMXnY0o>

<Https://www.youtube.com/watch?v=z2OagcHG-UU>

(cf also General Motors' earlier Token bus protocol)



Source: ProcessOnline.com

**SmartNIC:** NW Interface Card with general-purpose CPUs, to offload processing commonly done by server CPUs

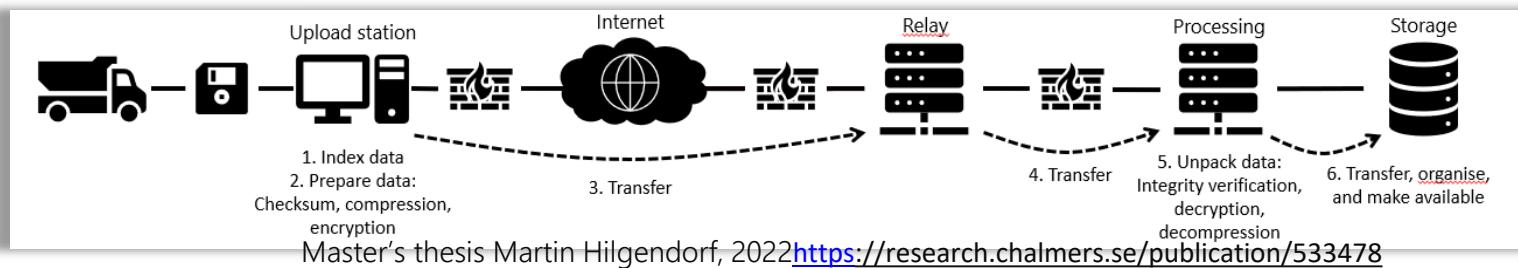
<https://community.arm.com/developer/ip-products/processors/b/processors-ip-blog/posts/reduce-tco-with-arm-based-smartnics>

<https://www.youtube.com/watch?v=WlwpfIUzRtQ>

more generalized programming: P4 (see p4.org)

IoT induces

- data pipeline (incl. processing @edge) issues



Master's thesis Martin Hilgendorf, 2022 <https://research.chalmers.se/publication/533478>

# Summary & Study list

## Limitations of classic Internet NW service model

Zoom in (again) inside a router: [Ch. 4.2.3-4.2.5]

- Decisions that influence timing/bandwidth guarantees (also related with congestion-control):  
Buffer management, packet scheduling, traffic shaping

## Dealing with limitations of destination-based forwarding

- Software-Defined Networking (SDN) / NW Function Virtualization (NFV)[ch 4.4, 5.5, 4.5]
  - Concept
  - Generalized Forwarding- OpenFlow
  - Ctrl Plane – OpenFlow
- Middleboxes, "Intelligence" and controls
- Internet-of-Things in evolution: more types of traffic/devices...



1. “Traditional” Internet NW core and transport protocol service models not suitable for streaming media traffic
2. Applications/edge took matters into own hands
  - New + evolving methods; new proposals for transport protocols
3. Another type of service @ core (VC-like) would imply a different situation
  - Internet core is re-shaping, for long time ... (Traffic engineering, SDN/NFV)
  - Any support of any new NW model requires **X-enabled routers**
4. **IoT in evolution: even more types of traffic, new needs**
  - efficient **in-NW processing /data pipelines** in focus.
  - check related research (incl. by our team, as well as Ericsson, Volvo, GE, ...)

---

**Extra slides/notes for further study and  
“food for thought”**

# Timing/bandwidth guarantees in networks:

## Tasks for the NW core (**think beyond/outside classic Internet**)

aka Quality of Service (QoS): 2-party agreement (NW user – NW provider) on

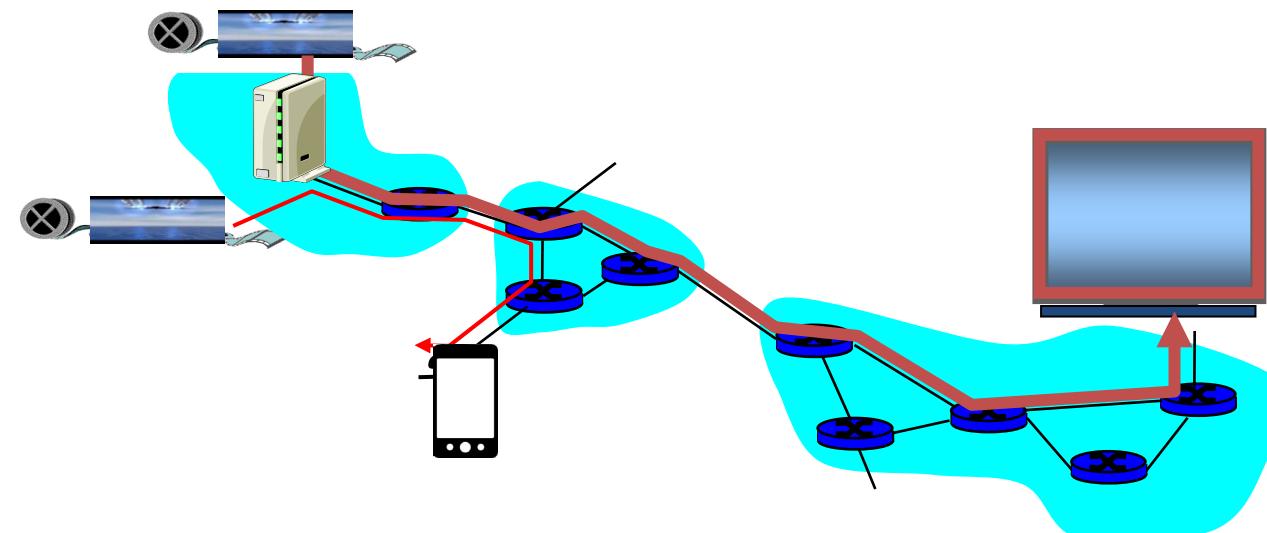
- Traffic characteristics (packet rate, sizes, ...)
- Network service guarantees (delay, jitter, loss rate, ...)

### *Resource sharing and congestion: questions/principles for QoS in NW Core*

- Distinguish: traffic & allocate: resources? (utilization)
- Control offered load? (isolate different "streams"?)

### *Tasks for the NW core:*

- Packet classification & scheduling (bandwidth allocation)
- Traffic shaping/policing (enforce contract terms)



# Difficulties with Datagram-based NW core...

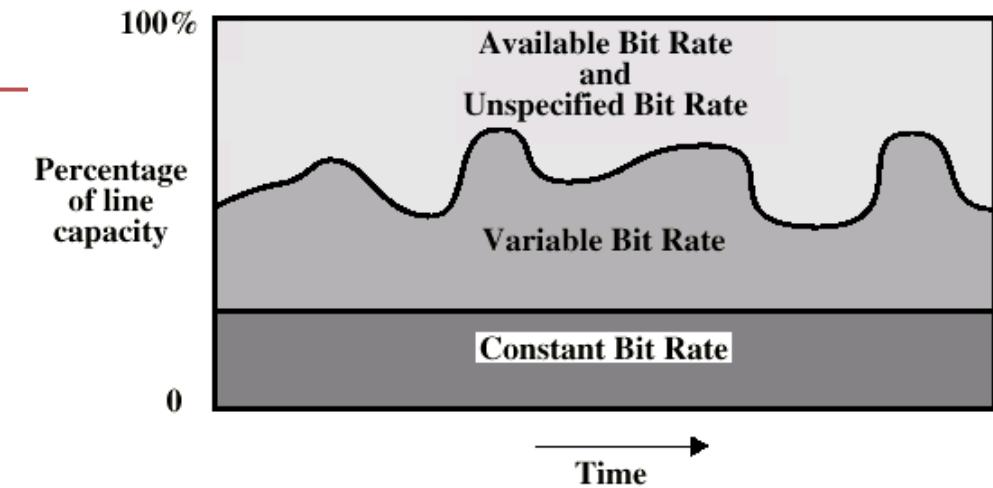
---

- No notion of "streams of packets per user/connection" to impose rate-limits...

# i.e. Internet NW service model does not specify traffic type *functionality*

Contrast with (earlier, not fully in-place) ATM Network service models: (telco proposed standards concurrent with proposal of TCP/IP)

| Service Model       | Example                  | Guarantees ?       |      |       |        | Congestion control |
|---------------------|--------------------------|--------------------|------|-------|--------|--------------------|
|                     |                          | Bandwidth          | Loss | Order | Timing |                    |
| Constant Bit Rate   | voice                    | constant rate      | yes  | yes   | yes    | Admission control  |
| VariableBR (RT/nRT) | Video/<br>“streaming”    | guaranteed rate    | yes  | yes   | yes    | Admission control  |
| Available BR        | WWW-browsing             | guaranteed minimum | no   | yes   | no     | Yes, feedback      |
| UndefinedBR         | Background file transfer | none               | no   | yes   | no     |                    |



# Virtual Circuit example: ATM: Asynchronous Transfer Mode nets

---

Internet's IP:

- today's *de facto* standard for global data networking

1980-1990's:

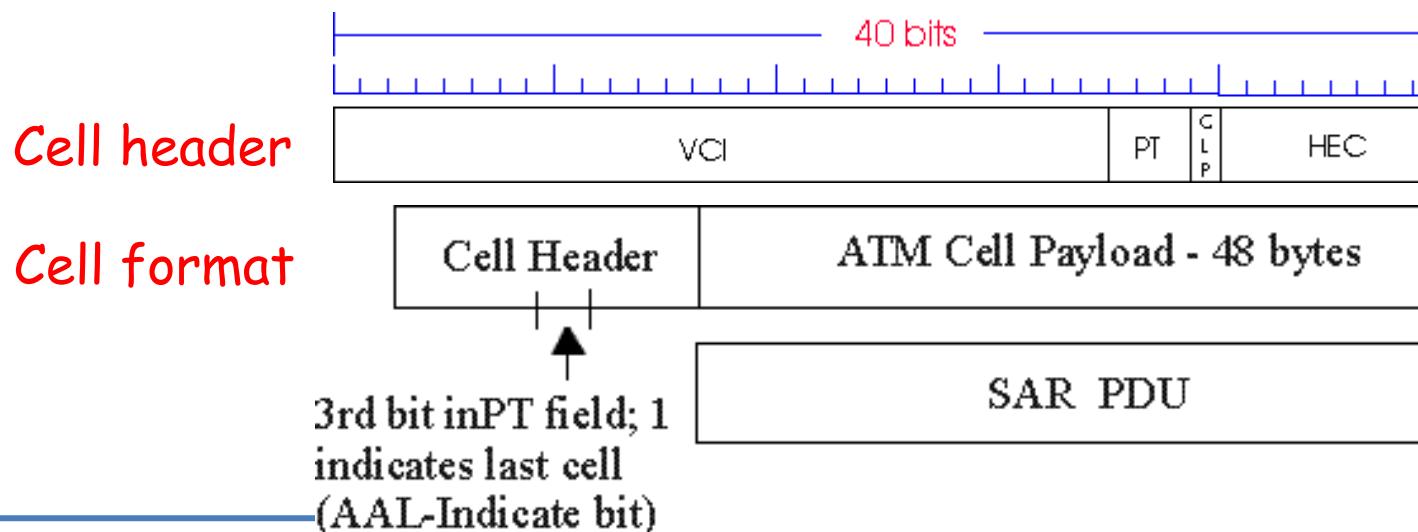
- telco's develop ATM specifications: competing network standard for carrying high-speed voice/data

ATM principles:

- **virtual-circuit networks**: switches maintain state for each "call"
- **Forwards on a per-flow basis**
- small (48 byte payload, 5 byte header) fixed length *cells* (like packets)
  - Ultra fast switching, directly on HW (almost no stop&fwd)
  - small size good for voice
- well-defined interface between "network" and "user" (think of classic, dumb telecom)

# ATM cell (small packet)

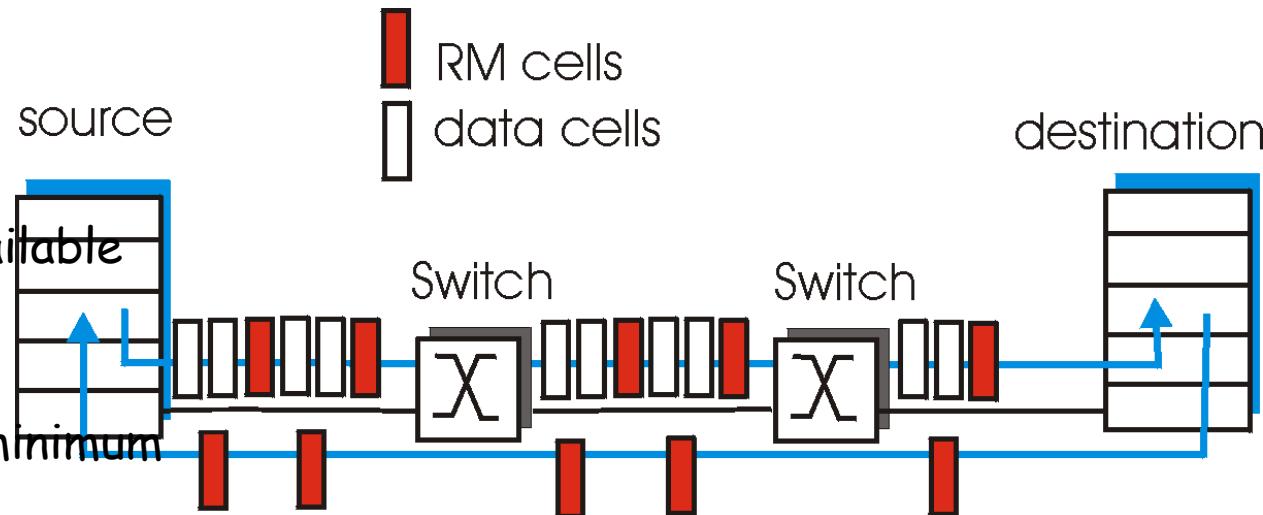
- 48-byte payload
  - Why?: small payload -> short cell-creation delay for digitized voice
  - halfway between 32 and 64 (compromise!)
- **Header: 5bytes**
  - **VCI**: virtual channel ID
  - **PT**: Payload type (e.g. Resource Management cell versus data cell)
  - **CLP**: Cell Loss Priority bit
    - CLP = 1 implies low priority cell, can be discarded if congestion
  - **HEC**: Header Error Checksum



# ATM ABR congestion control

ABR: available bit rate:

- r "elastic service"
- r if path "underloaded":
  - m sender should use available bandwidth
- r if path congested:
  - m sender throttled to minimum guaranteed rate



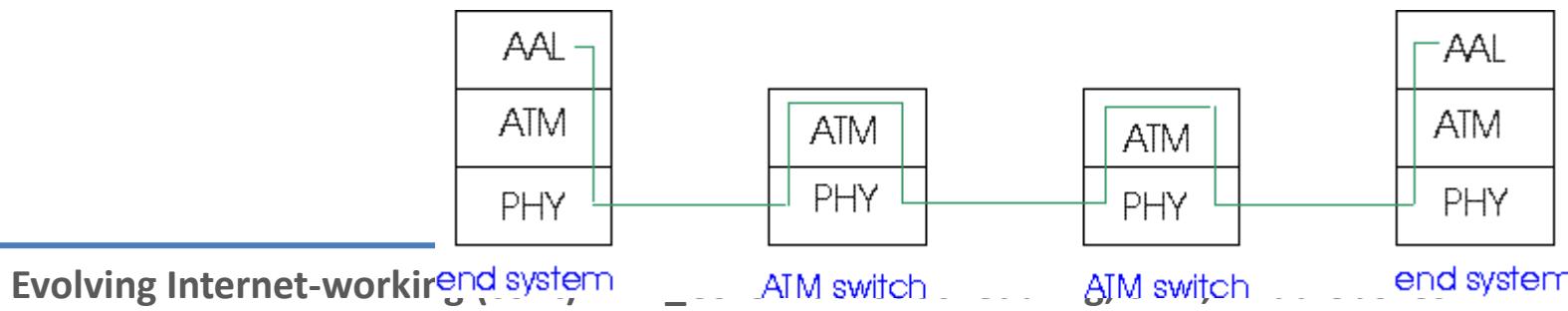
RM (resource management) cells:

- interspersed with data cells
- bits in RM cell set by switches ("network-assisted")
  - NI bit: no increase in rate (mild congestion)
  - CI bit: congestion indication two-byte ER (explicit rate) field in RM cell
  - congested switch may lower ER value in cell
  - sender's send rate thus minimum supportable rate on path

# ATM Adaptation (Transport) Layer: AAL

Basic idea: cell-based VCs need to be "complemented" to be supportive for applications.

- r Several ATM Adaptation Layer (AALx) protocols defined, suitable for different classes of applications
  - r AAL1: for CBR (Constant Bit Rate) services, e.g. circuit emulation
  - r AAL2: for VBR (Variable Bit Rate) services, e.g., MPEG video
  - r .....
- "suitability" has not been very successful
- computer science community introduced AAL5, (simple, elementary protocol), to make the whole ATM stack usable as switching technology for data communication under IP!



# Traffic Shaping and Policing in ATM

Enforce the QoS parameters: check if  
*Peak Cell Rate (PCR)* and *Cell Delay Variation (CDVT)* are within the negotiated limits:

**Generic Cell Rate Algo:** introduce:  
expected next time for a successive cell based on  $T = 1/\text{PCR}$   
**border time**  $L$  ( $= \text{CDVT}$ )  $< T$  in which next transmission may start (but never before  $T-L$ )

A **nonconforming cell** may be discarded or its *Cell Loss Priority* bit be set, so it may be discarded in case of congestion

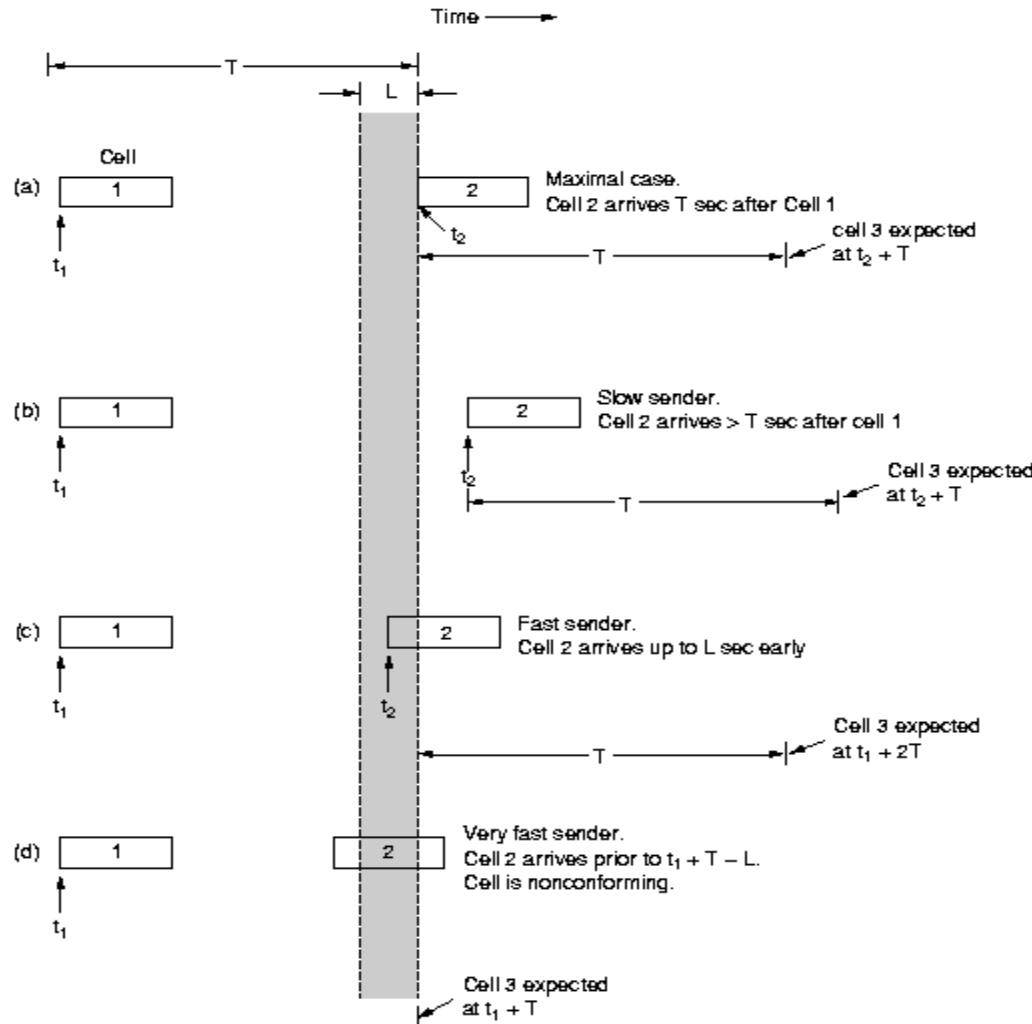


Fig. 5-73. The generic cell rate algorithm.

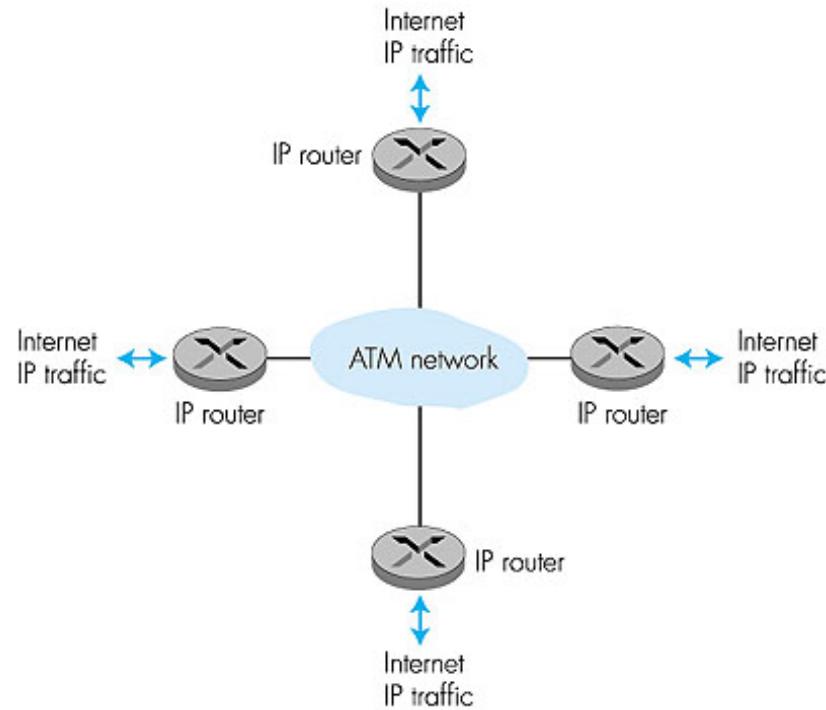
# What happened with ATM? network or link layer?

Vision: end-to-end transport: “ATM from desktop to desktop”

- ATM *is* a network technology

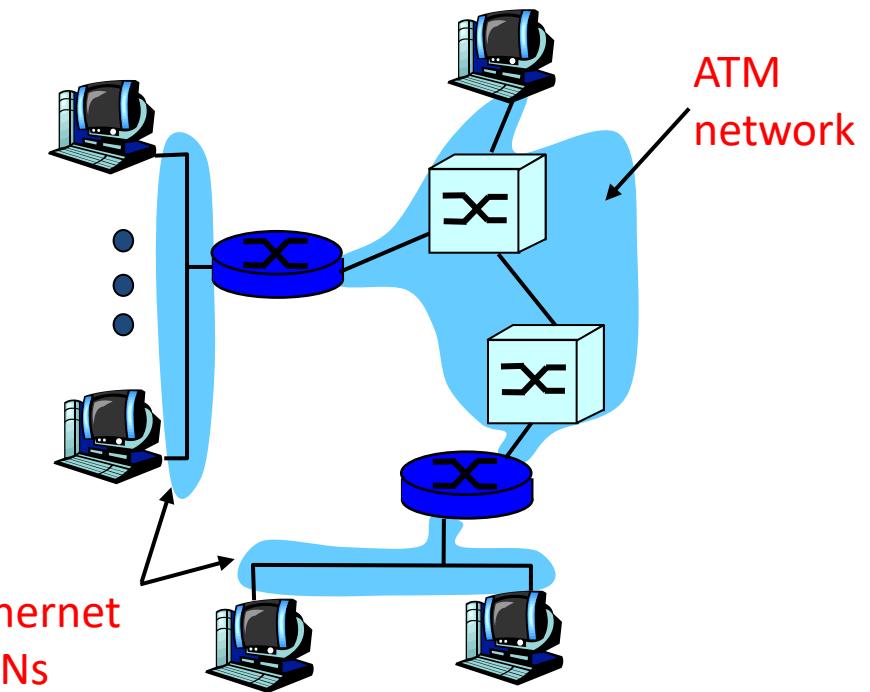
Reality:

- used to connect IP backbone routers ....

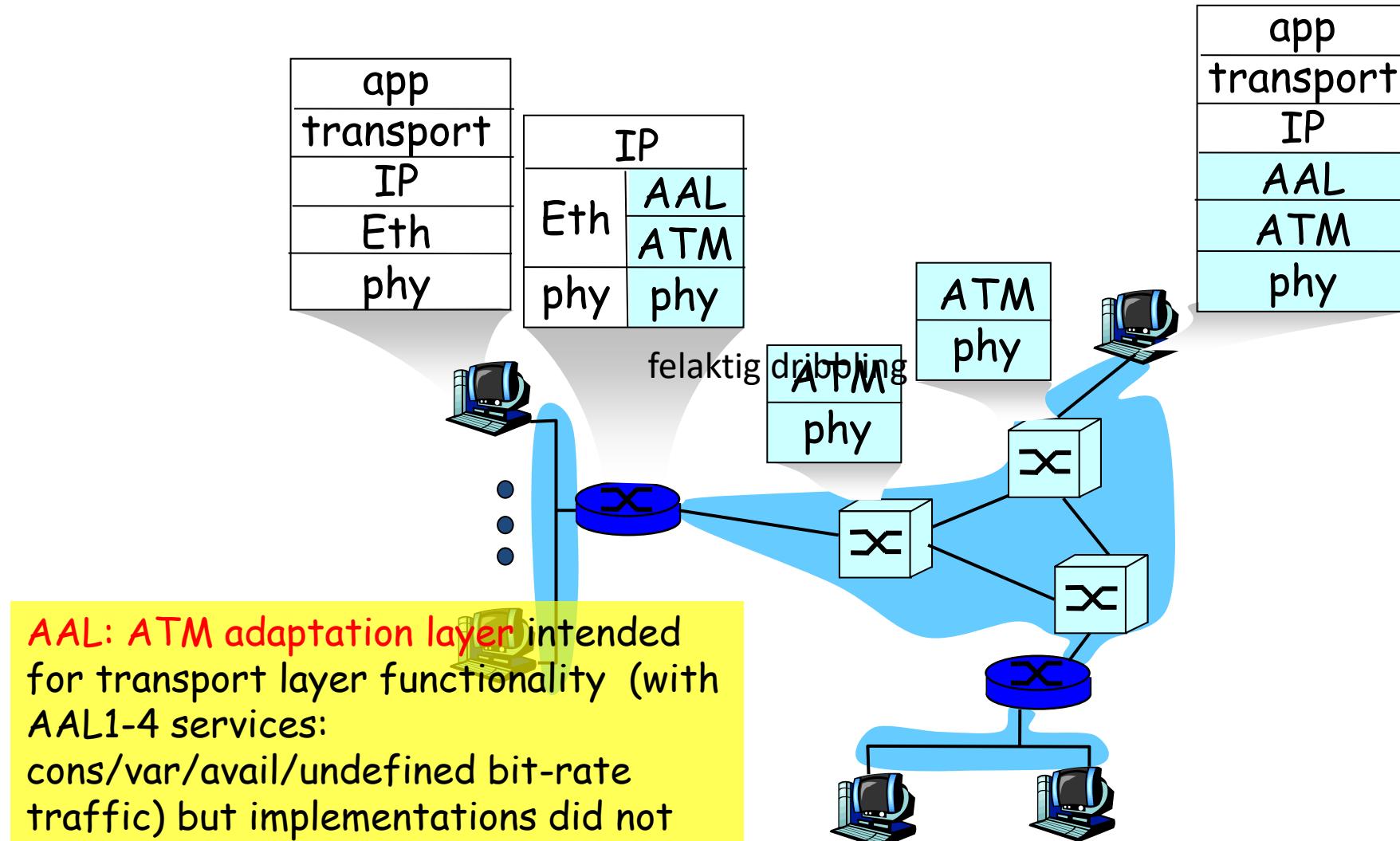


... or IP over ATM

- replace “network” (e.g., LAN segment) with ATM network, (ATM + IP addresses)
  - Run datagram routing on top of virtual-circuit routing!!!



# IP-Over-ATM

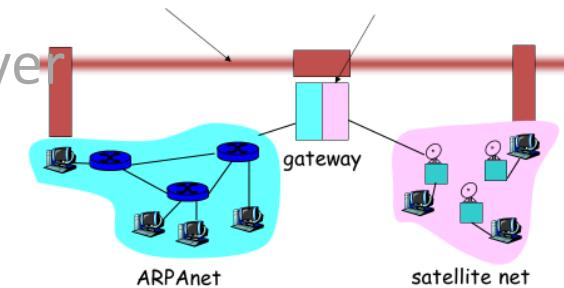


# ... ie. another instantiation of virtualization

## What is virtualized?

- two layers of addressing: internetwork and local network
- new layer (IP) makes everything homogeneous at internetwork layer
- underlying local network technology
  - Cable, satellite, ...
  - Ethernet, other LAN
  - ATM (expensive HW switches)
  - **Alternatively, do the same in SW: MPLS (Multiprotocol Label Switching Protocol): for traffic engineering**
- ... “invisible” at internetwork layer. Looks like a link layer technology to IP

Recall the Internet approach : virtualizing networks



M. Papatriantafilou - Evolving Internet-working Part B: NW\_Core: QoS, traffic engineering, SDN, IoT



# Examples rate-limiting methods: the effect of buckets

input

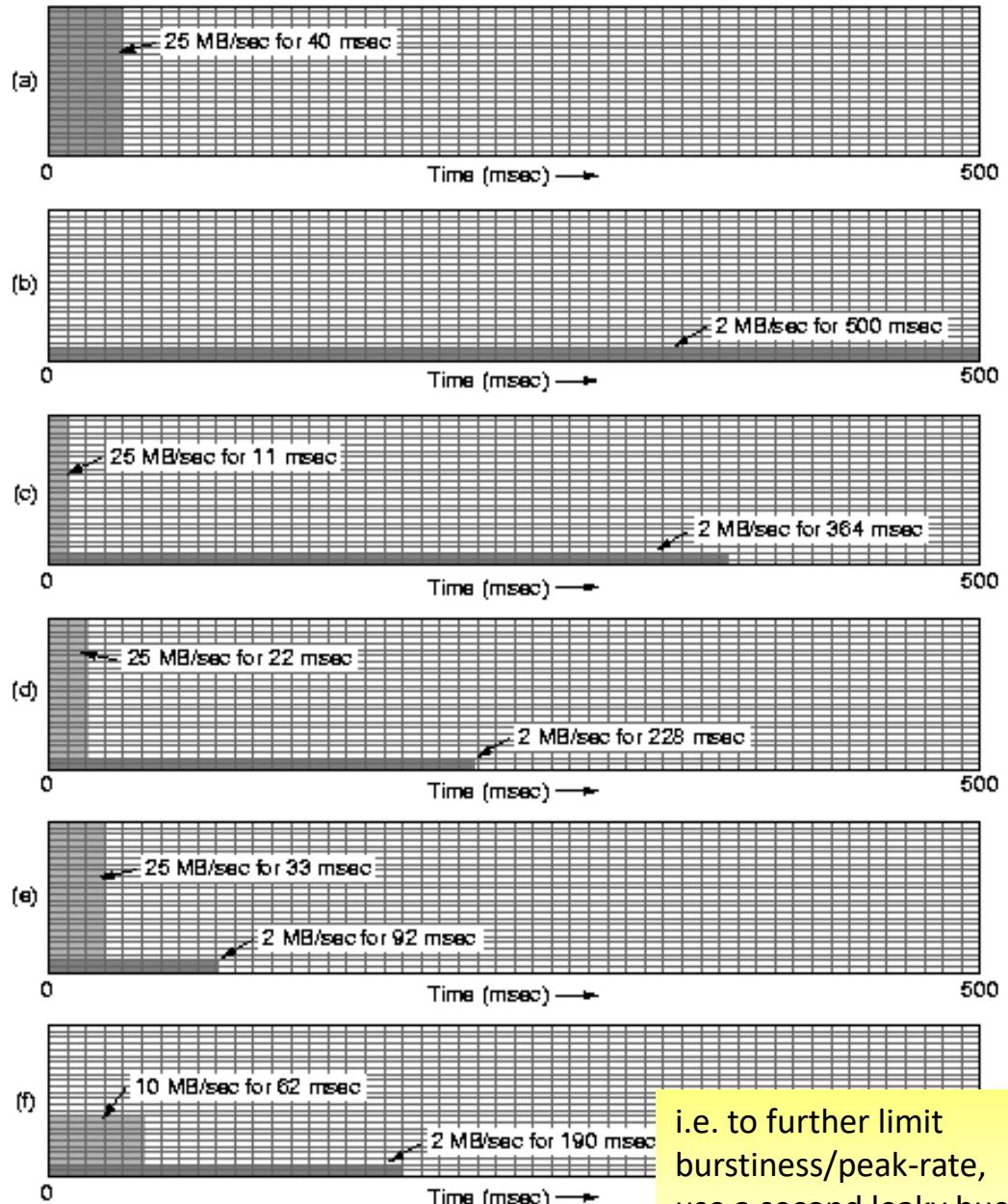
output 0KB token leaky bucket, 2MBps

output 250KB token leaky bucket, 2MBps

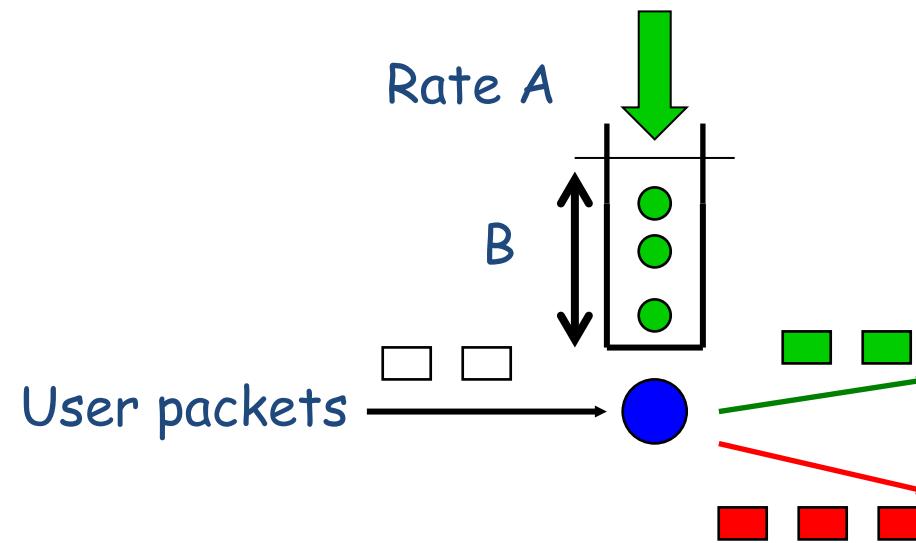
output 500KB token leaky bucket, 2MBps

output 750KB token leaky bucket, 2MBps

output token leaky bucket 500KB, 2MBps,  
feeding 0KB, 10MBps token leaky bucket



i.e. to further limit  
burstiness/peak-rate,  
use a second leaky bucket  
with higher rate



**-Class-based marking:** packets of different classes marked differently

**Profile within class:** pre-negotiated rate A, bucket size B

Packet is marked in the Type of Service (TOS) in IPv4, and Traffic Class in IPv6

**Forwarding:** according to “Per-Hop-Behavior” (PHB) strictly **based on classification marking**

- PHB **does not** specify mechanisms to ensure required PHB performance
- E.g.:
  - Class A gets  $x\%$  of bandwidth over time intervals of a specified length
  - Class A packets leave before class B packets

- **Advantage:**  
No state info to be maintained by routers

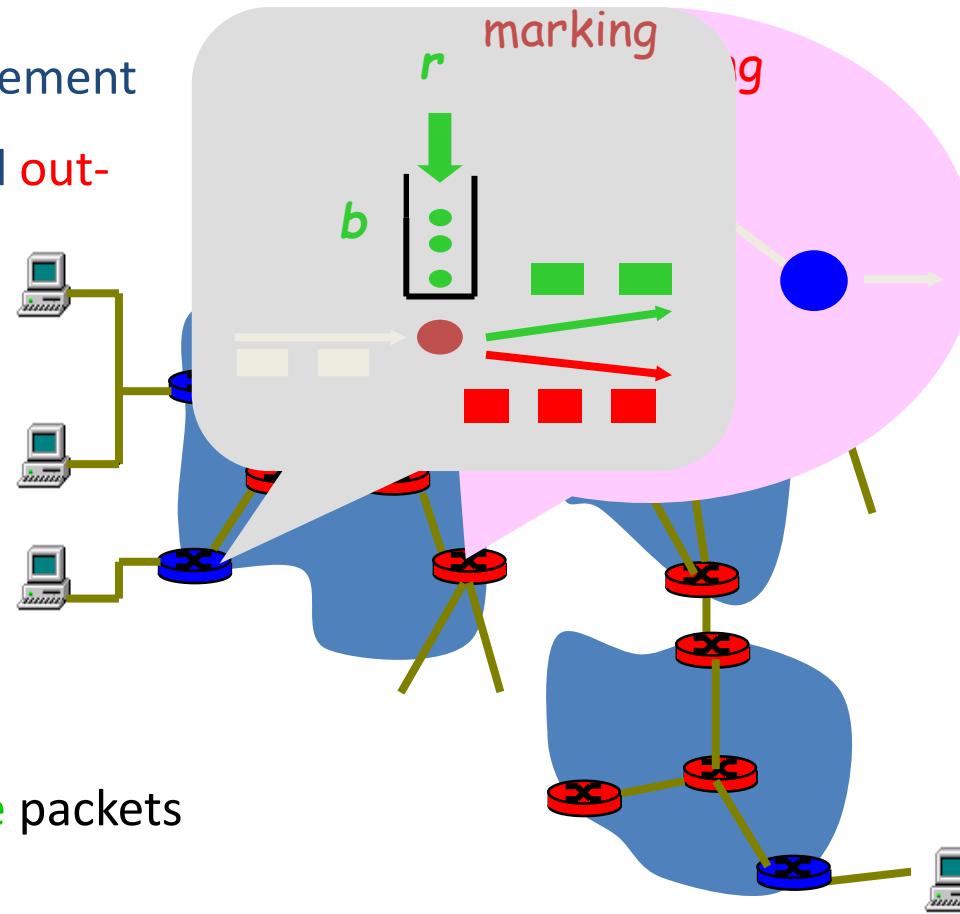
# Internet bandwidth-guarantee support possibilities?

## Diffserv proposed Architecture

### Edge router: marking



- per-class (=aggr-flow) traffic management
  - marks packets as **in-profile** and **out-of-profile**



### Core router: scheduling

- per class traffic scheduling
  - based on **marking** at edge
  - preference given to **in-profile** packets

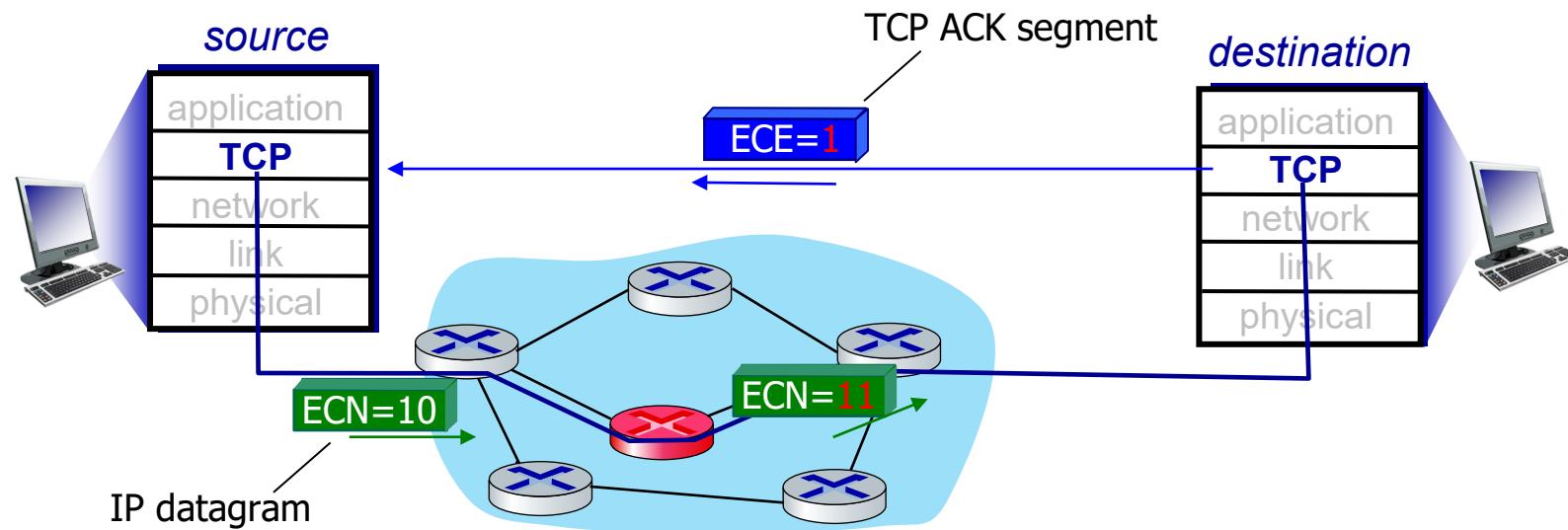
Provide functional components to build **service classes**

- Network core: stateless, simple
- Combine into **aggregated flows**, i.e. classification, shaping, admission: @ network edge

# About Marking and Explicit Congestion Notification (ECN)

Certain TCP deployments can implement *network-assisted* congestion control:

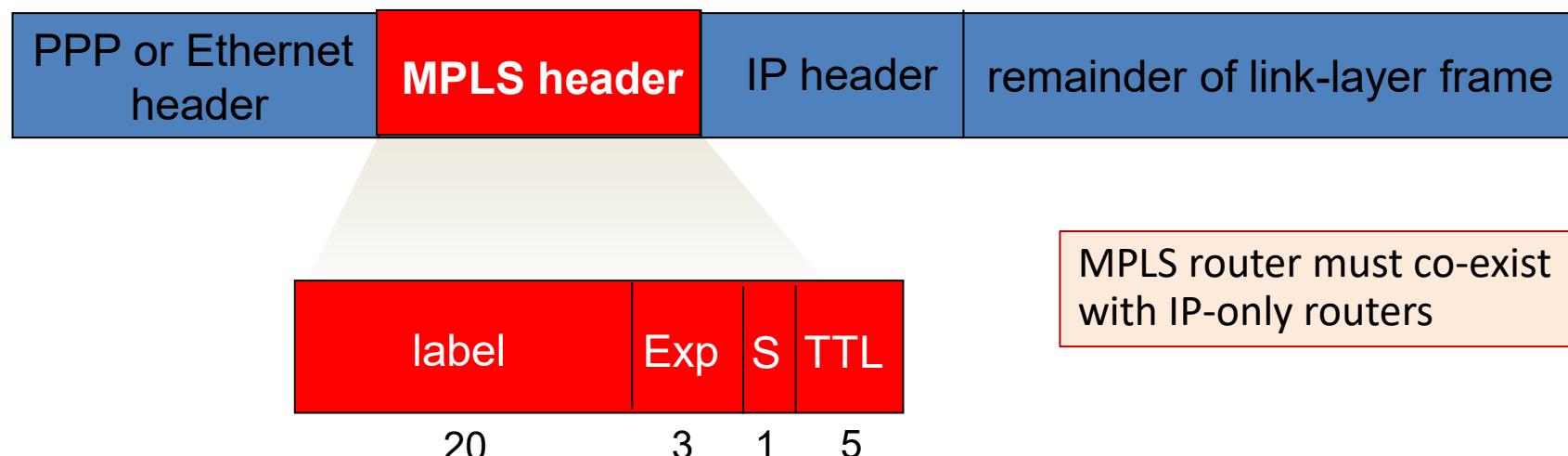
- two bits in IP header (ToS field) marked *by network router* to indicate congestion
  - policy to determine marking chosen by network operator
- congestion indication carried to destination
- destination sets ECE bit on ACK segment to notify sender of congestion
- involves both IP (IP header ECN bit marking) and TCP (TCP header C,E bit marking)



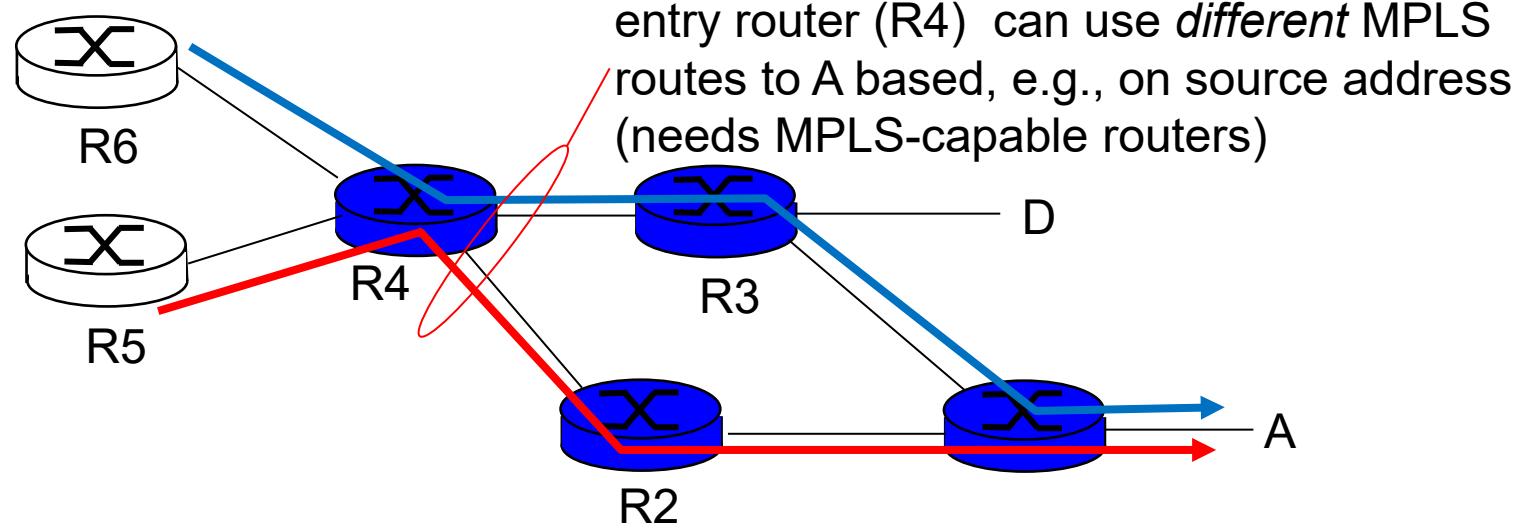


# Multiprotocol label switching (MPLS) in IP networks

- goal: utilize multiple S-T paths simultaneously
  - borrow ideas from Virtual Circuit (VC) approach
- MPLS': layer 2-3 protocol:
- Idea: label-based forwarding (label-switched router)
  - forwards packets to outgoing interface based only on label value (don't inspect IP address)
  - MPLS protocol's forwarding table distinct from IP forwarding tables



# MPLS versus IP paths



*IP routing:* path to destination determined by destination address alone



IP-only router

*MPLS routing:* path can be based on source *and* dest. address



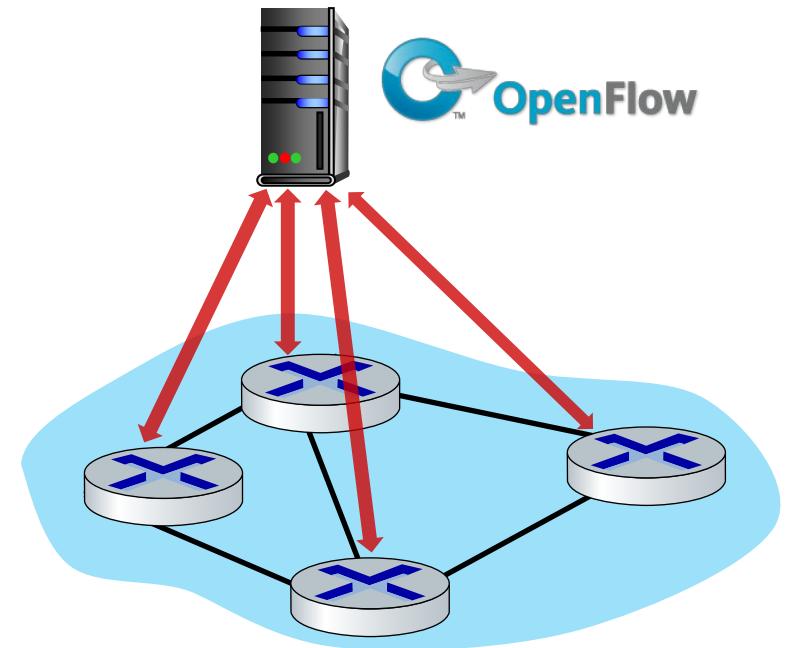
MPLS and IP router



# OpenFlow protocol

- operates between controller, switch
- TCP used to exchange messages
  - optional encryption
- three classes of OpenFlow Ctrl messages:
  - controller-to-switch
  - asynchronous (switch to controller)
  - symmetric (misc.)
- distinct from OpenFlow API
  - API used to specify generalized forwarding actions

OpenFlow Controller

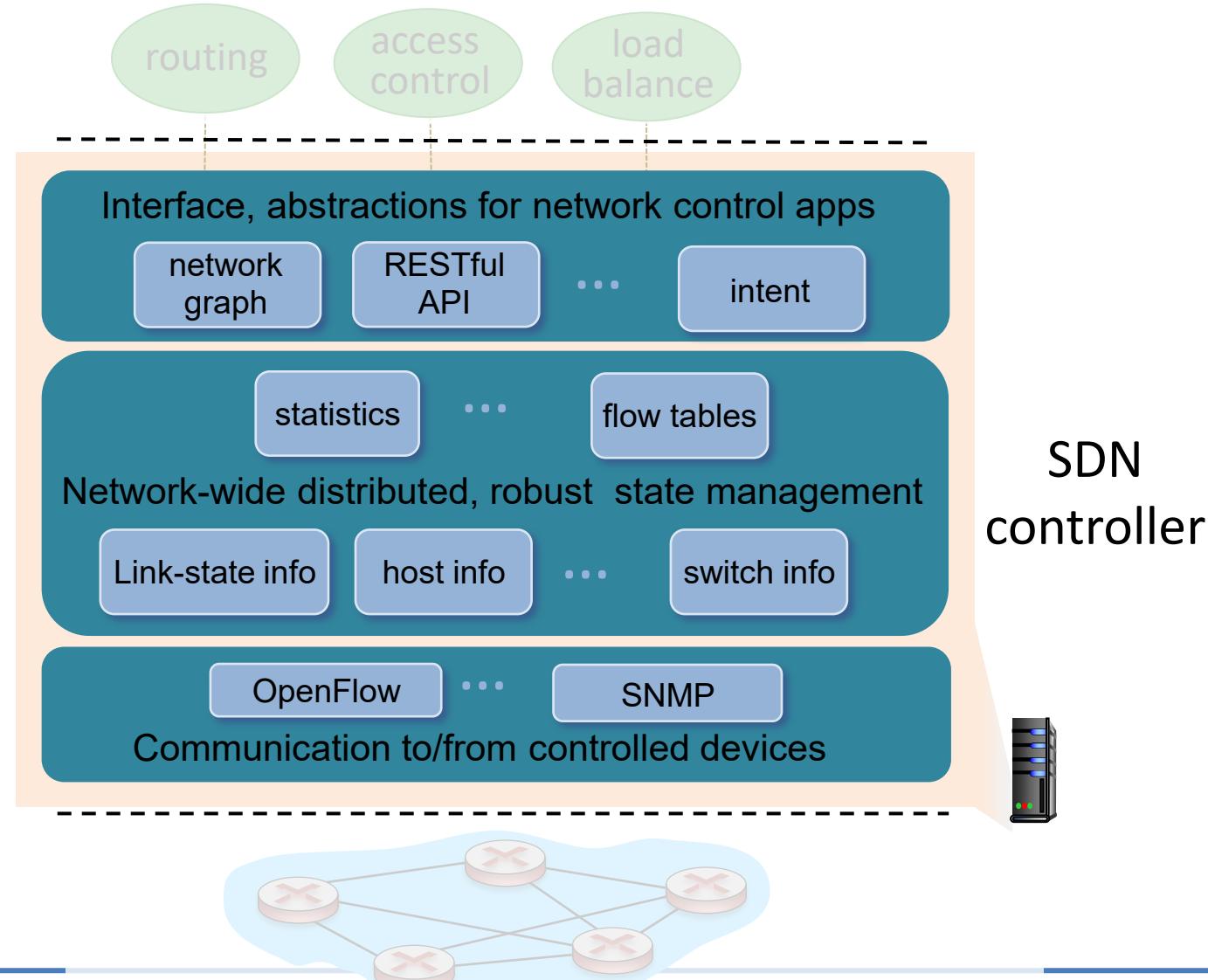


# Zooming in: components of SDN controller

Interface layer to network control apps: abstractions API

Network-wide state management layer: state of networks links, switches, services: a *distributed database*

*Communication layer:* communicate between SDN controller and controlled switches

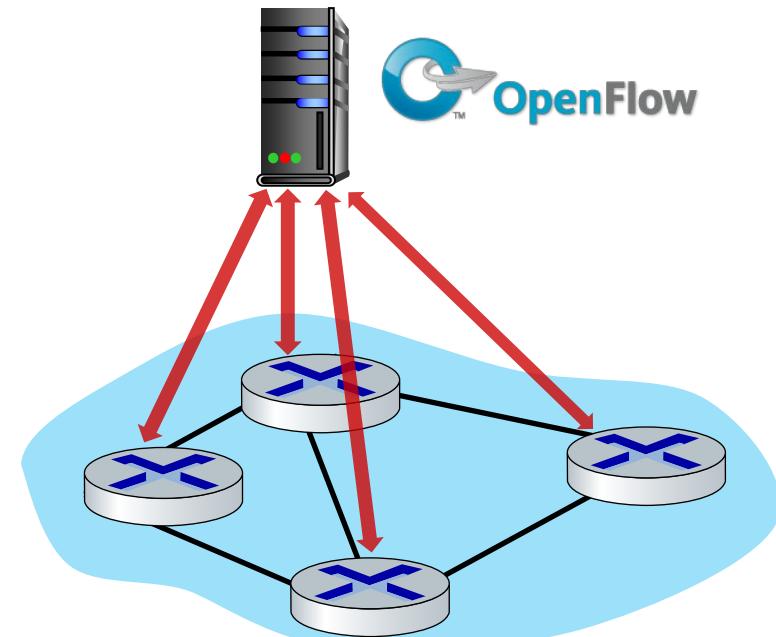


# OpenFlow: controller-to-switch messages

## Key controller-to-switch messages

- *features*: controller queries switch features, switch replies
- *configure*: controller queries/sets switch configuration parameters
- *modify-state*: add, delete, modify flow entries in the OpenFlow tables
- *packet-out*: controller can send this packet out of specific switch port

## OpenFlow Controller

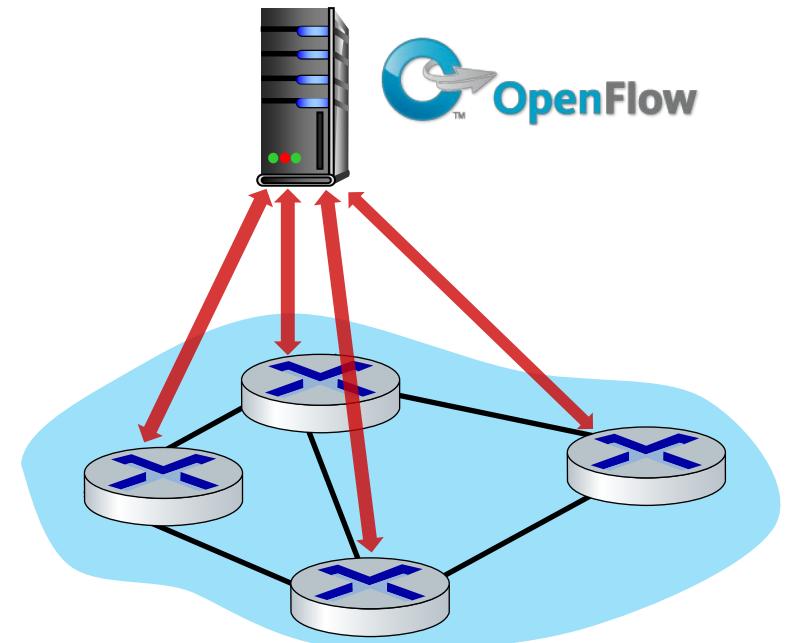


# OpenFlow: switch-to-controller messages

## Key switch-to-controller messages

- *packet-in*: transfer packet (and its control) to controller. See packet-out message from controller
- *flow-removed*: flow table entry deleted at switch
- *port status*: inform controller of a change on a port.

## OpenFlow Controller



Fortunately, network operators don't "program" switches by creating/sending OpenFlow messages directly. Instead use higher-level abstraction at controller

# Generalized forwarding: summary

---

- “match plus action” abstraction: match bits in arriving packet header(s) in any layers, take action
  - matching over many fields (link-, network-, transport-layer)
  - local actions: drop, forward, modify, or send matched packet to controller
  - “program” *network-wide* behaviors
- simple form of “network programmability”
  - programmable, per-packet “processing”
  - *historical roots*: active networking
  - *today*: more generalized programming:  
P4 (see p4.org), programmable/smart NIC



# Middleboxes

---

- initially: proprietary (closed) hardware solutions
- move towards “whitebox” hardware implementing open API
  - move away from proprietary hardware solutions
  - programmable local actions via match+action
  - move towards innovation/differentiation in software
- SDN: (logically) centralized control and configuration management often in private/public cloud
- network functions virtualization (NFV): programmable services over white box networking, computation, storage

# Architectural Principles of the (early) Internet

## RFC 1958

“Many members of the Internet community would argue that there is no architecture, but only a tradition, which was not written down for the first 25 years (or at least not by the IAB). However, in very general terms, the community believes that

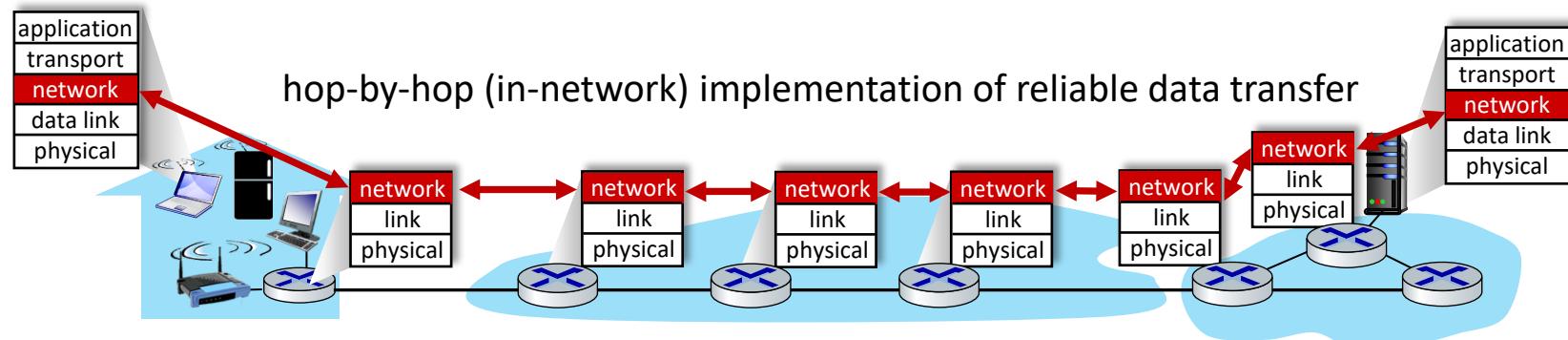
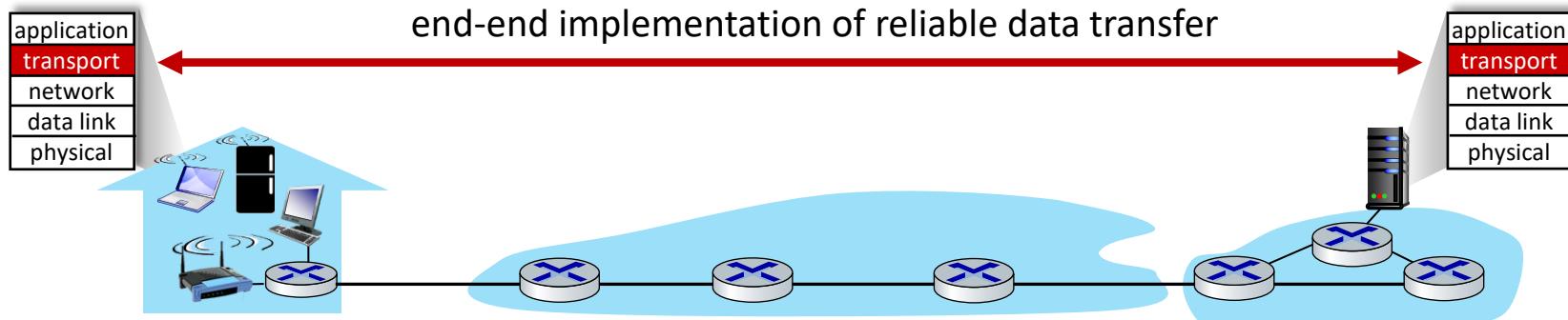
**the goal is connectivity, the tool is the Internet Protocol, and the intelligence is end to end rather than hidden in the network.”**

Three cornerstone beliefs:

- simple connectivity
- IP protocol: that narrow waist
- intelligence, complexity at network edge

# The end-end argument

- network functionality (requiring extra “work”/“intelligence” e.g., reliable data transfer, congestion avoidance) can/should be implemented **network edge (instead of in-network)**



See Saltzer, Reed, Clark  
ACM TOCS 1984

# The end-end argument

---

- some network functionality (e.g., reliable data transfer, congestion) can be implemented in network, or at network edge

“The function in question can completely and correctly be implemented only with the knowledge and help of the application standing at the end points of the communication system. Therefore, providing that questioned function as a feature of the communication system itself is not possible. (Sometimes an incomplete version of the function provided by the communication system may be useful as a performance enhancement.)

We call this line of reasoning against low-level function implementation the “end-to-end argument.”

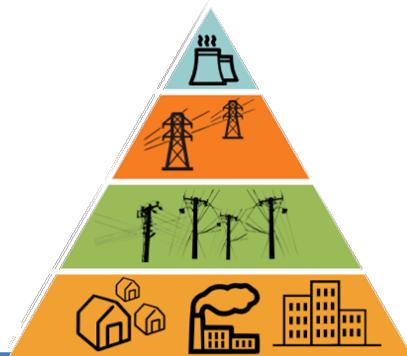
---

Saltzer, Reed, Clark ACM TOCS 1984



# Data networking technologies in Smart Grids

Presentation by  
Giorgos Georgiadis  
(former CTH / curr Volvo AB)



# Introduction

1

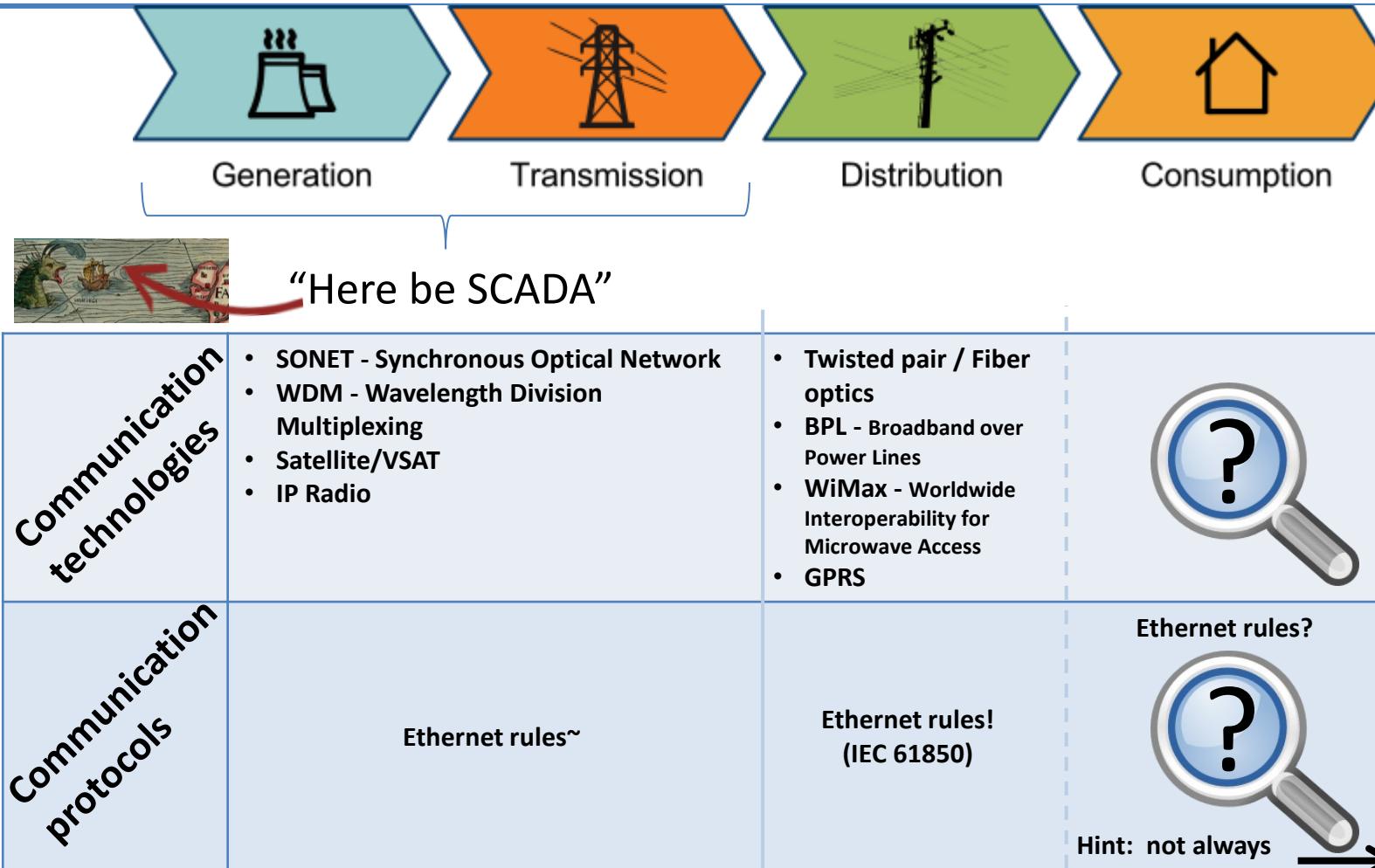
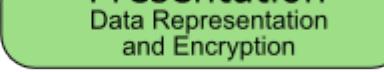
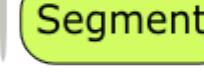
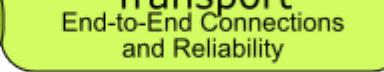
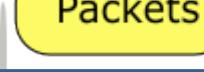
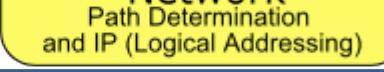
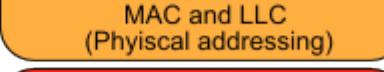
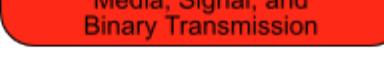


Fig. Giorgos Georgiadis

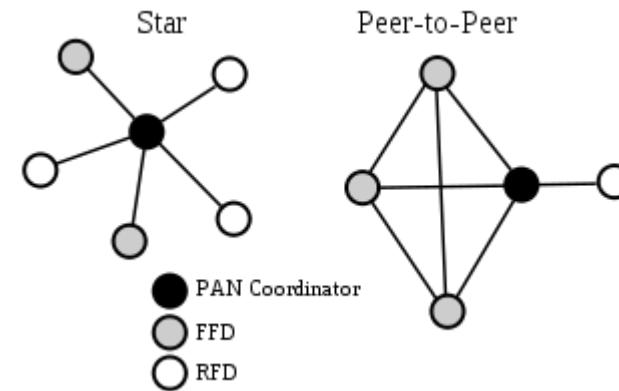
# Approximate overview of shaping new stacks

| OSI Model    |                                                                                                                                                                                                                                                                                                                                                                                                            | Protocols @ Distribution's last mile |                                                                                                 |               |                               |
|--------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|--------------------------------------|-------------------------------------------------------------------------------------------------|---------------|-------------------------------|
|              | Data Layer                                                                                                                                                                                                                                                                                                                                                                                                 |                                      |                                                                                                 |               |                               |
| Host Layers  | <div style="display: flex; justify-content: space-around;"> <div style="text-align: center;">  <p>Data</p> </div> <div style="text-align: center;">  <p>Application<br/>Network Process to Application</p> </div> </div>                | OpenADR<br>REST-based (i.e. CoAP)    |                                                                                                 |               |                               |
|              | <div style="display: flex; justify-content: space-around;"> <div style="text-align: center;">  <p>Data</p> </div> <div style="text-align: center;">  <p>Presentation<br/>Data Representation and Encryption</p> </div> </div>           | XMPP<br>BACNet<br>LonWorks<br>Modbus | ZigBee                                                                                          | 6LoW PAN      | HomePlug                      |
| Media Layers | <div style="display: flex; justify-content: space-around;"> <div style="text-align: center;">  <p>Segments</p> </div> <div style="text-align: center;">  <p>Transport<br/>End-to-End Connections and Reliability</p> </div> </div>      |                                      |  <p>WiFi</p> |               |                               |
|              | <div style="display: flex; justify-content: space-around;"> <div style="text-align: center;">  <p>Packets</p> </div> <div style="text-align: center;">  <p>Network<br/>Path Determination and IP (Logical Addressing)</p> </div> </div> |                                      |                                                                                                 |               |                               |
|              | <div style="display: flex; justify-content: space-around;"> <div style="text-align: center;">  <p>Frames</p> </div> <div style="text-align: center;">  <p>Data Link<br/>MAC and LLC (Physical addressing)</p> </div> </div>         | Proprietary                          | Ethernet / Gigabit Ethernet                                                                     | IEEE 802.15.4 | Proprietary, part 2: HomePlug |
|              | <div style="display: flex; justify-content: space-around;"> <div style="text-align: center;">  <p>Bits</p> </div> <div style="text-align: center;">  <p>Physical<br/>Media, Signal, and Binary Transmission</p> </div> </div>       |                                      |                                                                                                 |               |                               |

- Ethernet
  - Not much to say
- HomePlug
  - Honorable mention: popular home automation protocol
  - Powerline based
  - Speed: ~200mbps
  - Otherwise, vanilla protocol:
    - i.e. using TDMA,
    - Two kinds of nodes,
    - ...

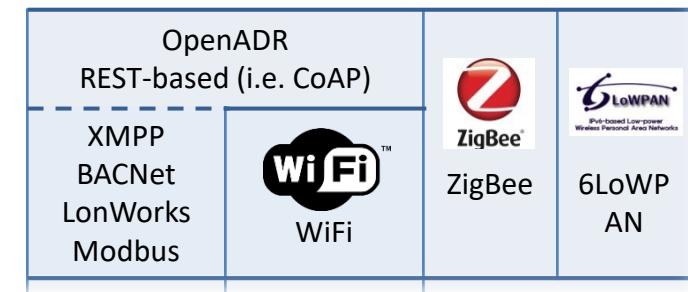
|             |                             |               |                               |
|-------------|-----------------------------|---------------|-------------------------------|
| Proprietary | Ethernet / Gigabit Ethernet | IEEE 802.15.4 | Proprietary, part 2: HomePlug |
|-------------|-----------------------------|---------------|-------------------------------|

- IEEE 802.15.4
  - Radio based, usually 2.4GHz
  - Small packets (<=127bytes)
  - Medium speed (~250kbps)
  - Originally DSSS
  - Topologies supported:
    - Star
    - Peer-to-peer
  - Roles supported:
    - Full-function device
    - Reduced-function device

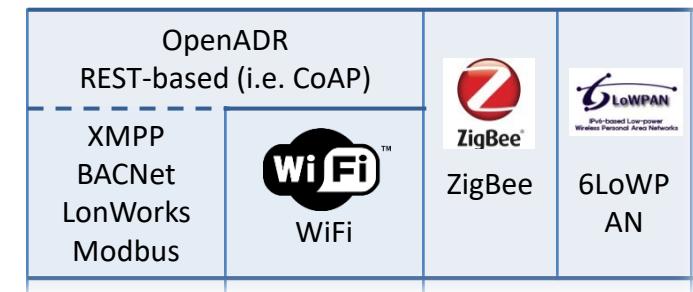


| Proprietary | Ethernet / Gigabit Ethernet | IEEE 802.15.4 | Proprietary, part 2: HomePlug |
|-------------|-----------------------------|---------------|-------------------------------|
|-------------|-----------------------------|---------------|-------------------------------|

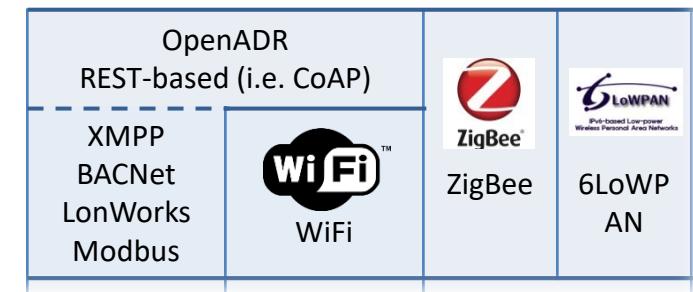
- 6LoWPAN
  - “IPv6 over LoW Power wireless Area Networks”
  - Builds on 802.15.4, IPv6
  - Aimed at low power devices (sensors, controllers)
  - Topologies
    - Star, peer-to-peer + Mesh
  - Many Challenges:
    - IP packets  $\geq 1280$ bytes (!)
    - 128bit IP addresses
    - ...



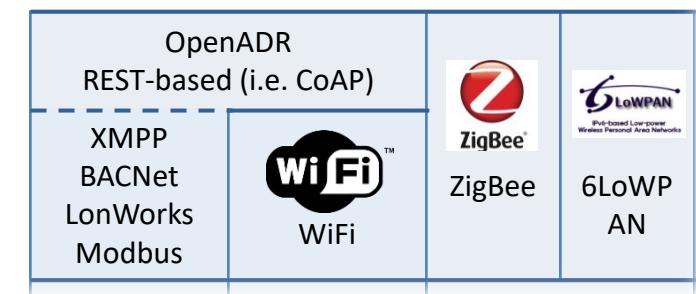
- ZigBee
  - Builds on 802.15.4, **but not IP**
  - Aimed at low power devices too (sensors, controllers)
    - Speed 250kbps
    - Packet 127bytes
    - Battery powered devices (supports sleep)
  - Topologies supported
    - + Mesh (jump to: example)
  - Roles supported
    - Coordinator, router, end node
  - Different profiles exist:
    - ZigBee Home Automation
    - Zigbee Smart Energy
    - **Zigbee IP**, ...



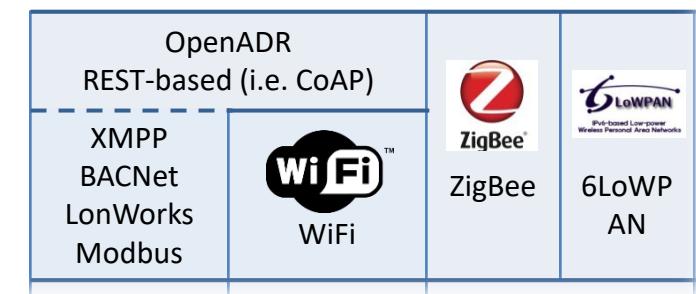
- More protocols, same story:
  - XMPP, BACNet, LonWorks, Modbus, ...
  - Wired
  - Proprietary, build around specific companies (BACNet, LonWorks) or legacy protocols (Modbus)
  - Today gateway devices to “break out” to Ethernet are in use
  - Simple topologies (i.e bus), same roles as before
- But what is the connecting thread over all?
  - Open standards!
  - Internet! (of Things?)



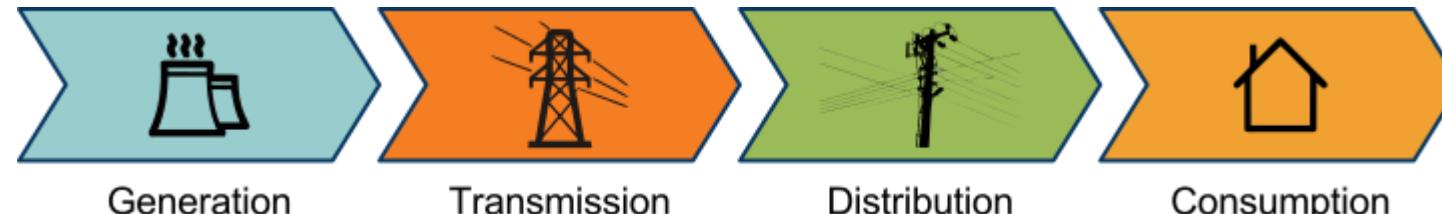
- OpenADR
  - ADR: Advanced Metering Response
  - Trying to ‘unify’ different solutions in a high level protocol
  - Formalizing:
    - Roles
    - Messages
    - Device detection
  - Simple topologies (i.e bus), same roles as before
- REST-based APIs
  - I.e. Costrained Application Protocol
  - Ultimately, HTTP-based
  - Verb oriented: GET, PUT, DELETE, ...



- Ethernet/IP-based integration
  - Remember:
    - Radio band: 2.4GHz (WiFi, ZigBee, 6LoWPAN)
    - Similar topologies, roles
    - Made for low energy devices, but flops/watt/kr increase!
    - Ethernet gateways commonly used
  - Solution: make them (formally) interoperable
    - ZigBee Smart Energy v2.0
    - ZigBee, WiFi, HomePlug on board
    - 6LoWPAN coming soon



# Conclusion



- Ethernet + misc communication technologies
- Ethernet vs non-ethernet
  - Why?
    - Design for low energy devices (smaller packets, lower comm speed)
    - Peer to peer, mesh topologies
  - Now + Future?
    - Devices' specs catching up
    - Importance of being connected (to the Internet?)
    - Topologies still important (i.e. reliability)
    - Will probably remain radio-based



# Course on Computer Communication and Networks

## Lecture 12 Chapter 7; Wireless Networks

EDA344/DIT423/LEU062

Lecturer: Hans-Martin Heyn

Based on the book Computer Networking: A Top Down Approach, Jim Kurose, Keith Ross, Addison-Wesley.

# What is coming next?



- Wireless Networking



- Security



- New Ideas: Evolving Networks

# Guest Lecture on Mon. 4th March at 10:00 **VOLVO**

- **Mahboobeh Daftari** is a Security Architect (PKI Lead Architect) at Volvo Trucks, Göteborg.
  - She has worked as Security Expert in the automotive industry for more than 6 years.
- She graduated from Chalmers in 2016 with a M.Sc. in Computer Systems and Networks

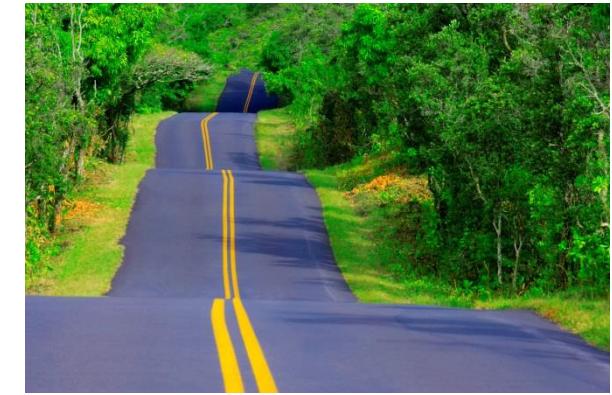


“Communication between vehicles and OEM backend systems is required for many purposes including incident analysis and vehicle software update. A common way to secure end-to-end communication between Electronic Control Units (ECUs) and OEM backend systems is to secure diagnostic service requests and responses between diagnostic tools and the ECUs.

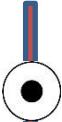
**Unified Diagnostic Services (UDS) protocol is a standard protocol defining two services for enabling security of diagnostics on the application layer:** Security Access (0x27) and Authentication Service (0x29). The overall security of the communication depends on the implementation of the services and the underlying key management which is OEM specific.”

# Roadmap Data-Link Layer

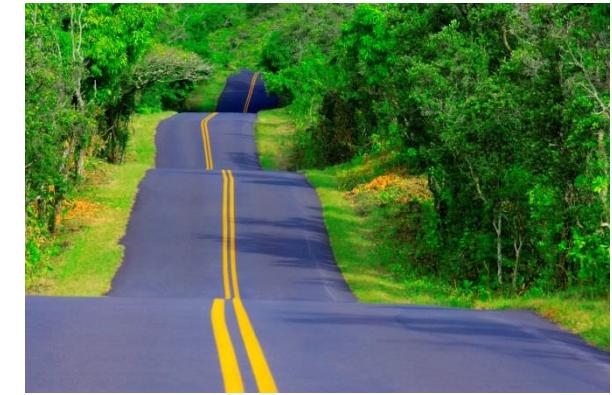
- Introduction to Wireless Networks
- Wireless Links and Network Characteristics
- WiFi: 802.11 WLANs
- Cellular Networks: 4G and 5G
- Mobility management
  - Principles
  - Practice
  - Impact on higher-layer protocols



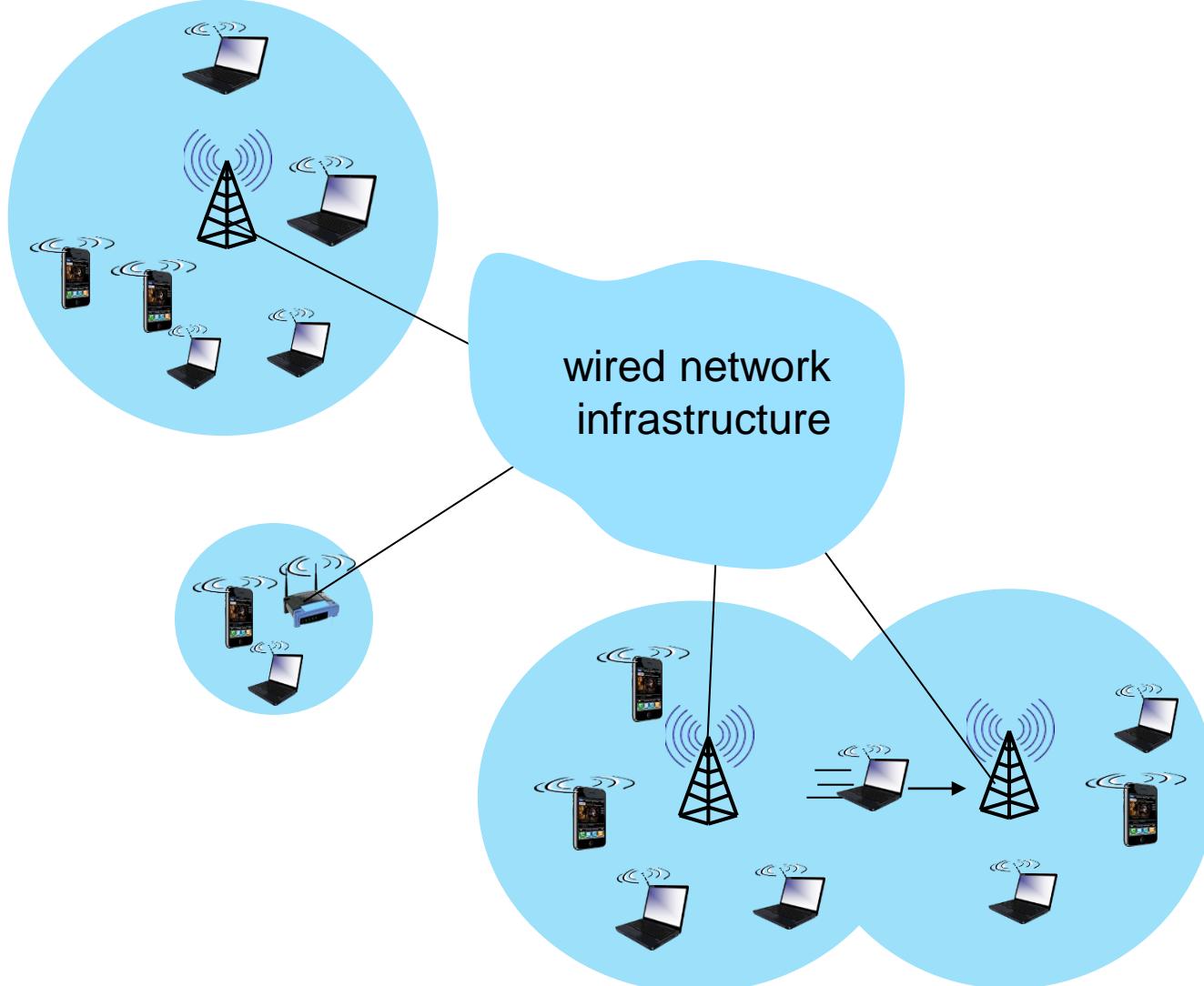
# Roadmap Data-Link Layer



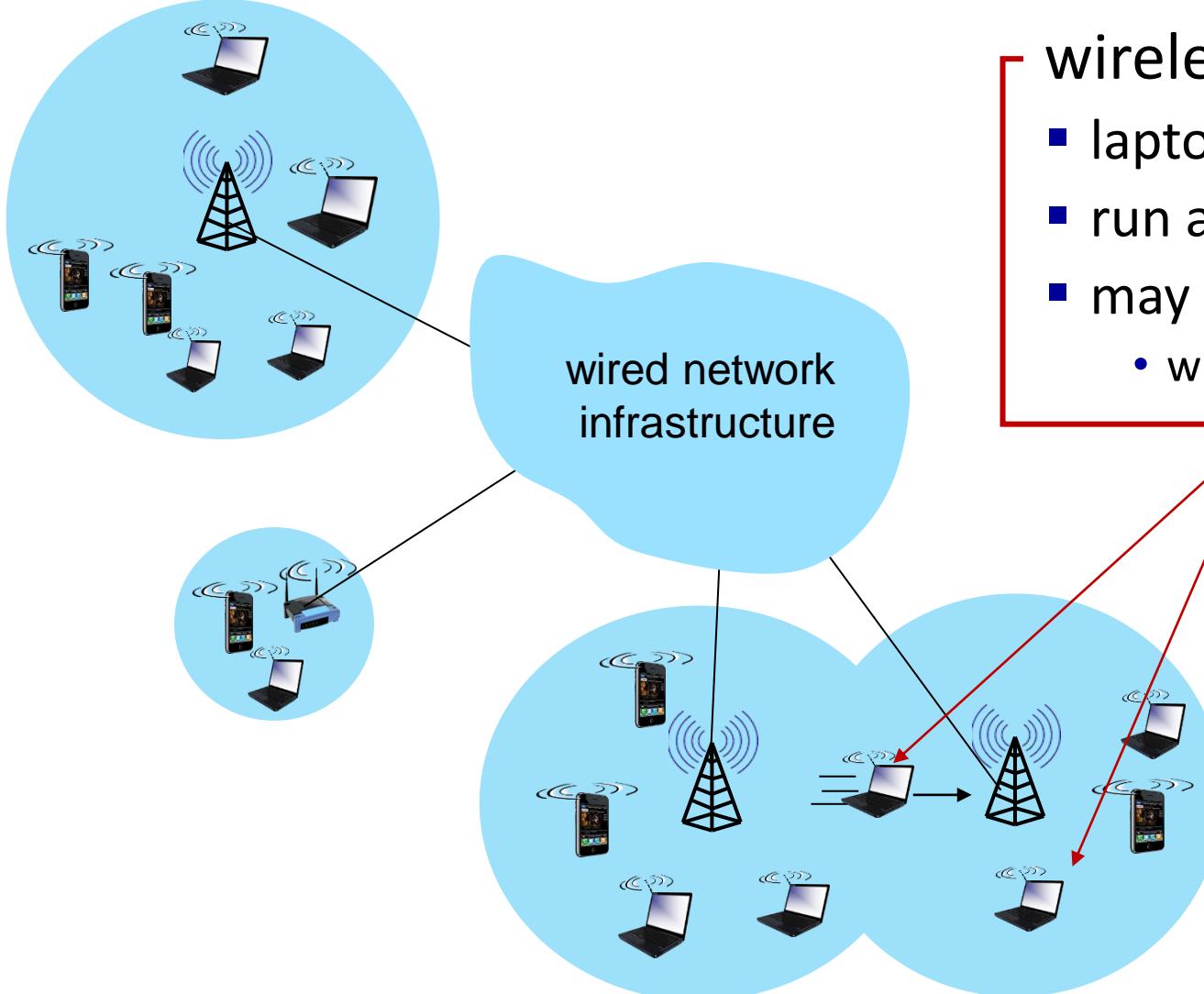
- Introduction to Wireless Networks
  - Wireless Links and Network Characteristics
  - WiFi: 802.11 WLANs
  - Cellular Networks: 4G and 5G
  - Mobility management
    - Principles
    - Practice
    - Impact on higher-layer protocols



# Elements of a wireless network



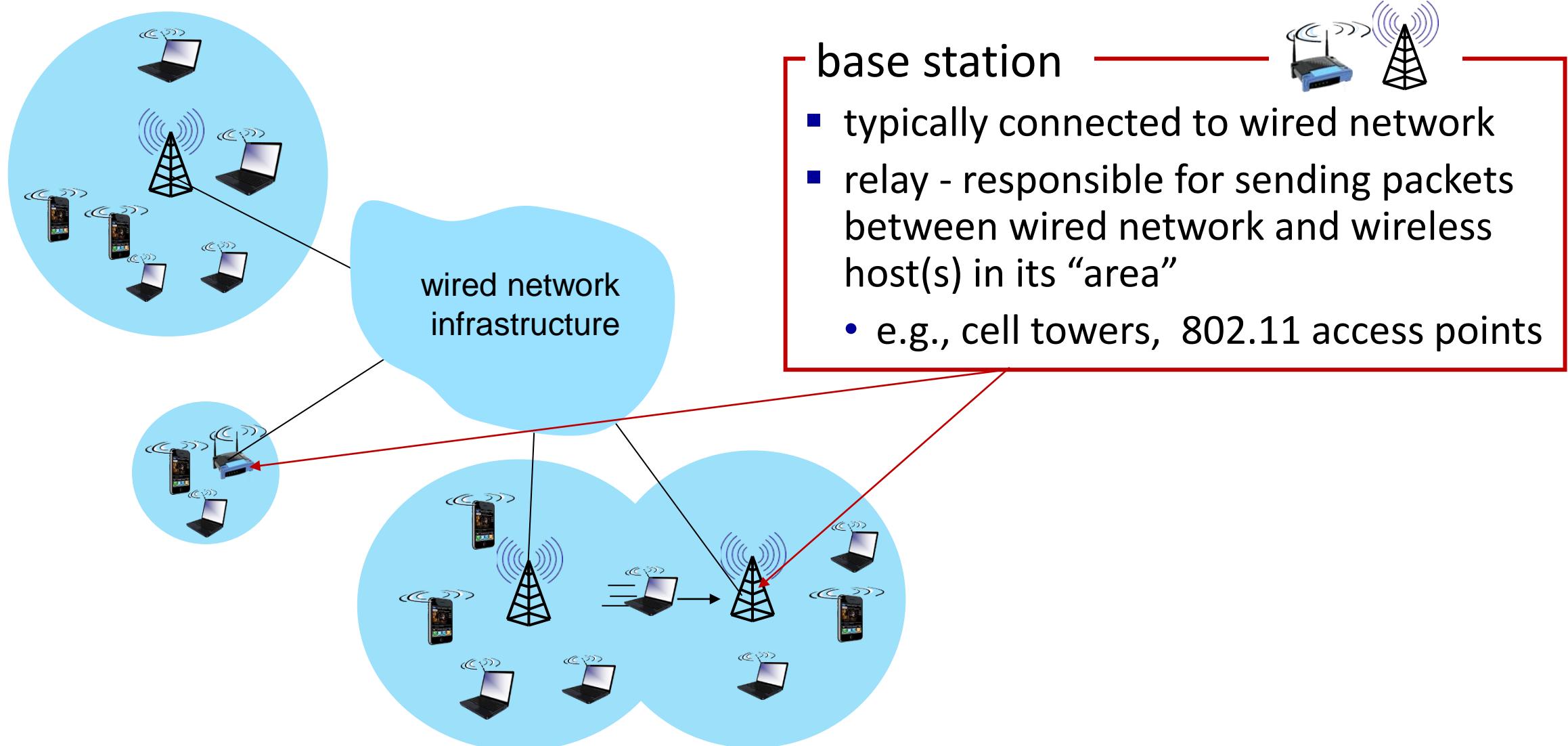
# Elements of a wireless network



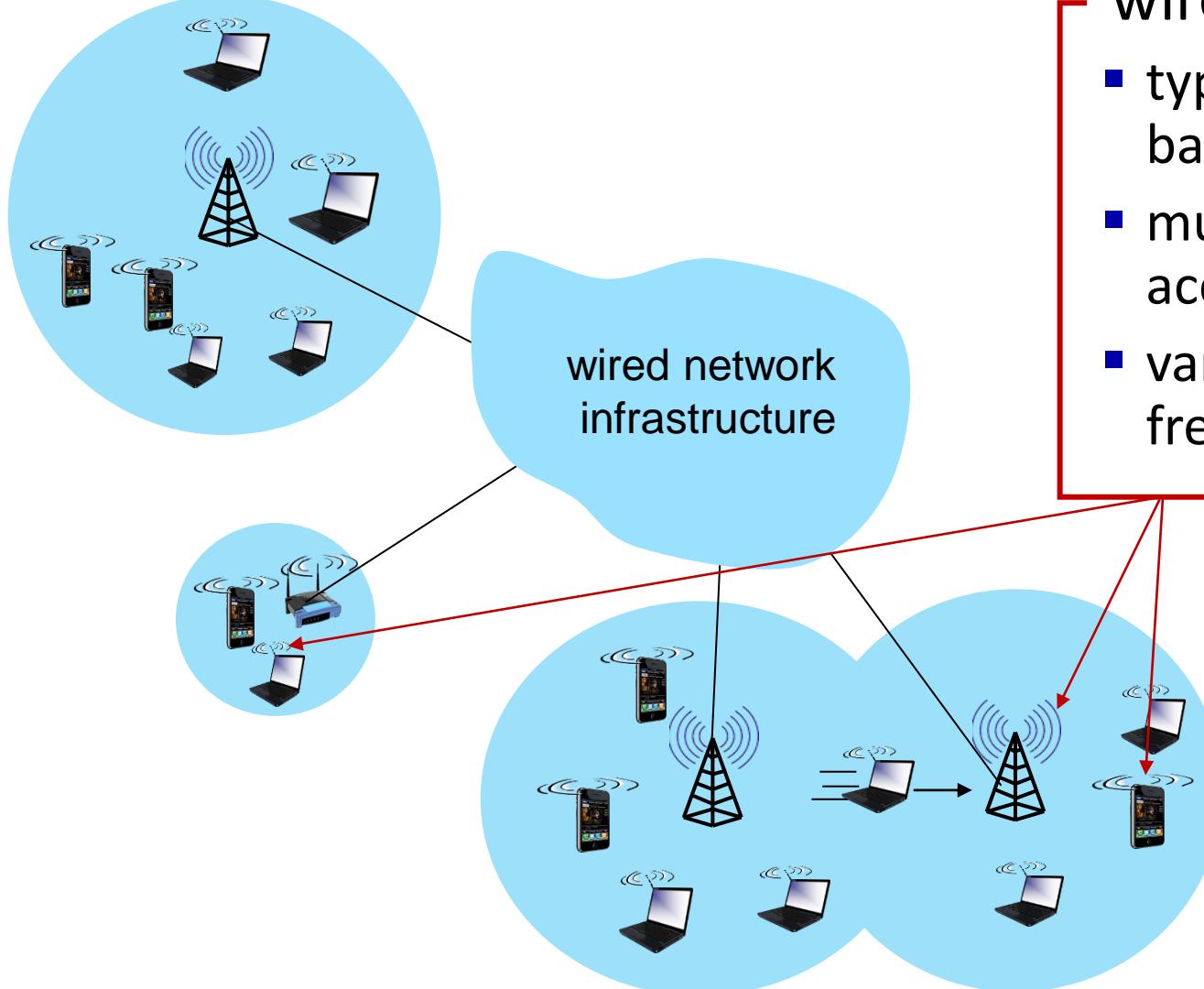
## wireless hosts

- laptop, smartphone, IoT
- run applications
- may be stationary (non-mobile) or mobile
  - wireless does *not* always mean mobility!

# Elements of a wireless network



# Elements of a wireless network

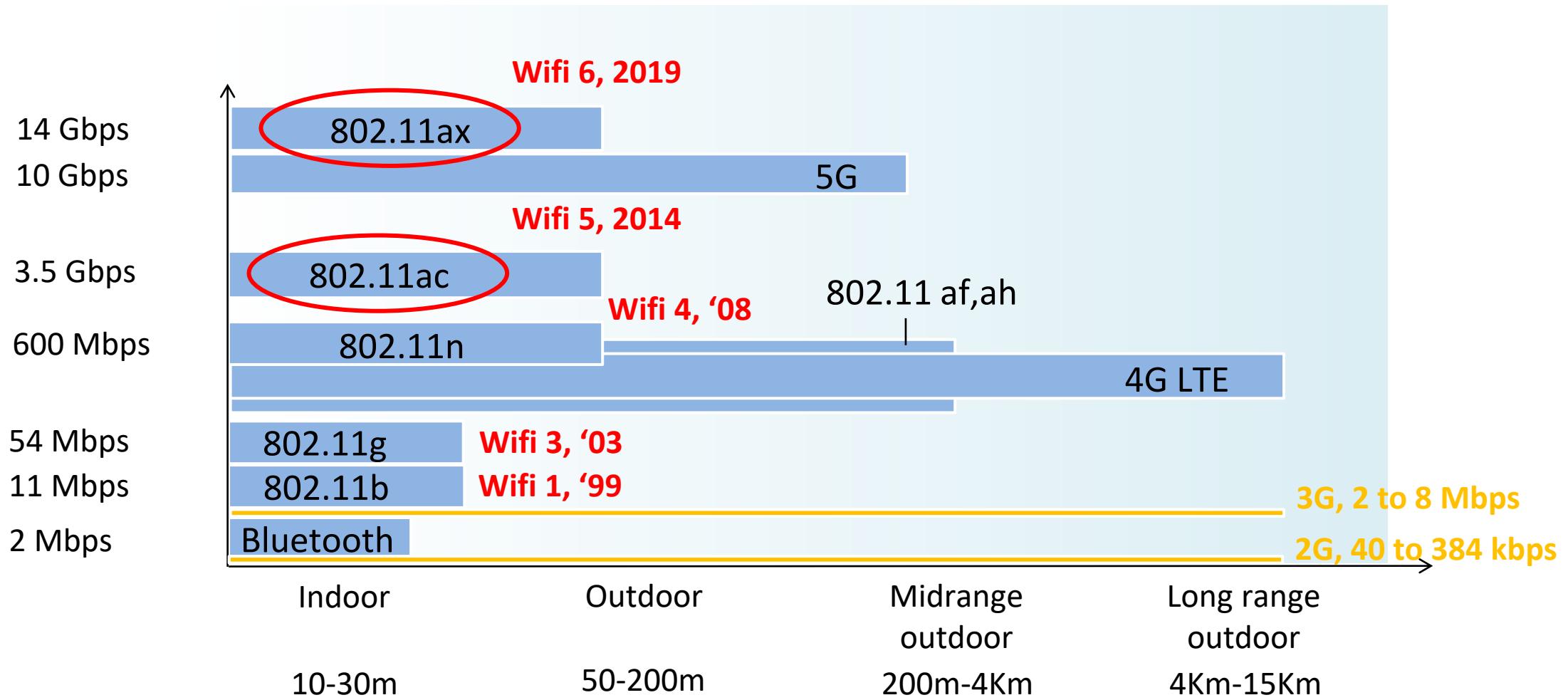


## wireless link

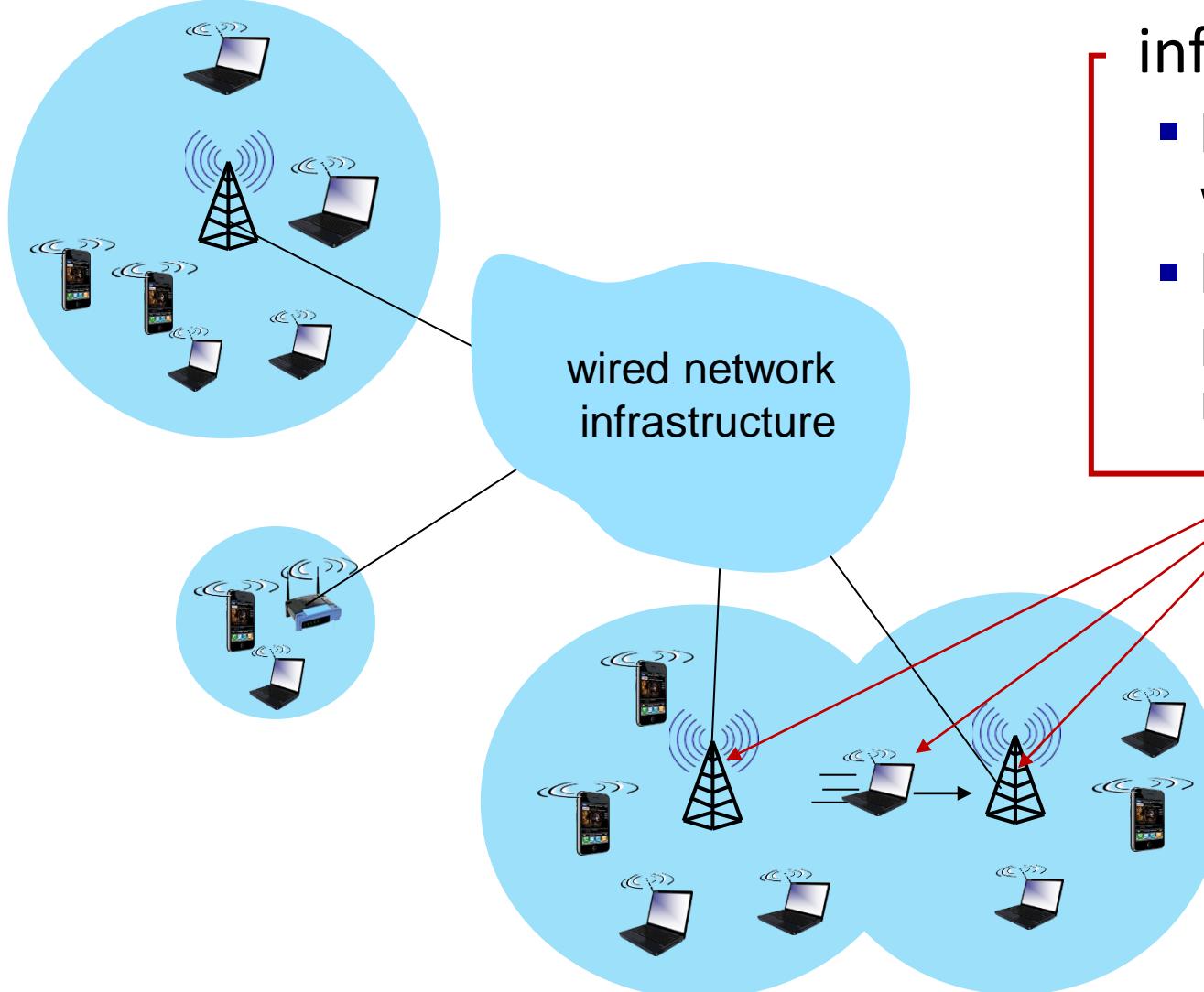
- typically used to connect mobile(s) to base station, also used as backbone link
- multiple access protocol coordinates link access
- various transmission rates and distances, frequency bands



# Characteristics of selected wireless links



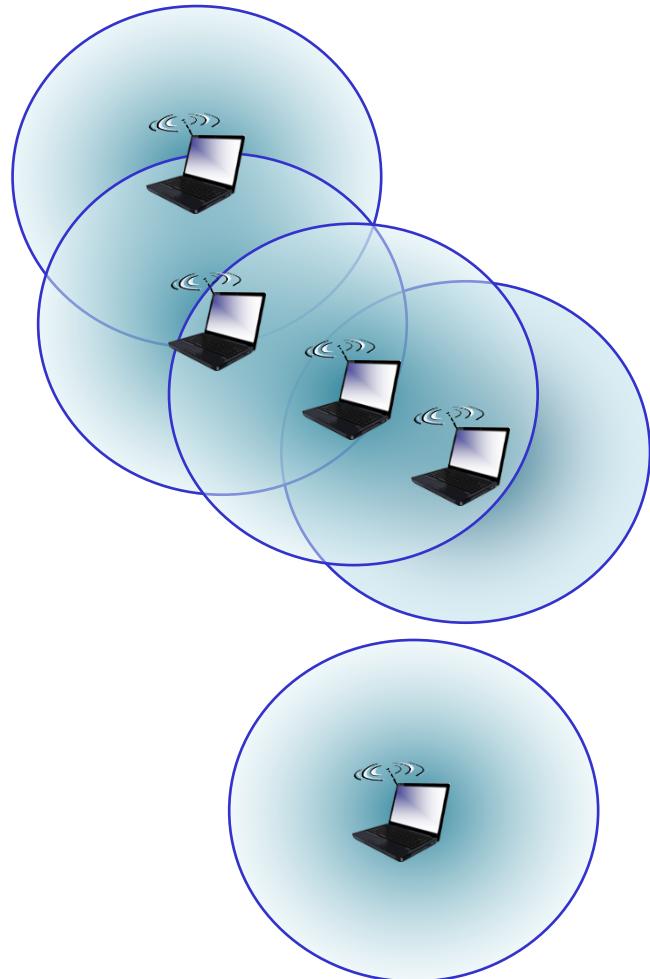
# Elements of a wireless network



## infrastructure mode

- base station connects mobiles into wired network
- handoff: mobile changes base station providing connection into wired network

# Elements of a wireless network



ad hoc mode

- no base stations
- nodes can only transmit to other nodes within link coverage
- nodes organize themselves into a network: route among themselves

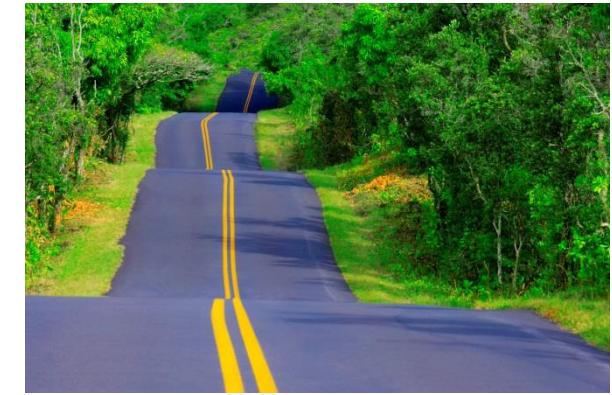
# Wireless network taxonomy

|                               | single hop                                                                                 | multiple hops                                                                                              |
|-------------------------------|--------------------------------------------------------------------------------------------|------------------------------------------------------------------------------------------------------------|
| infrastructure<br>(e.g., APs) | host connects to base station ( <i>WiFi, cellular</i> ) which connects to larger Internet. | host may have to relay through several wireless nodes to connect to larger Internet: <i>mesh net</i> .     |
| <i>no infrastructure</i>      | no base station, no connection to larger Internet ( <i>Bluetooth, ad-hoc nets</i> ).       | no base station, no connection to larger Internet. May have to relay to reach other a given wireless node. |

# Roadmap Data-Link Layer



- Introduction to Wireless Networks
- Wireless Links and Network Characteristics
- WiFi: 802.11 WLANs
- Cellular Networks: 4G and 5G
- Mobility management
  - Principles
  - Practice
  - Impact on higher-layer protocols



# The Maxwell-Heaviside equations

---

$$\nabla \cdot \mathbf{E} = \rho \varepsilon_0^{-1}$$

Divergence of the electric field is proportional to local density of electric charge

$$\nabla \cdot \mathbf{B} = 0$$

There are no magnetic monopoles!

$$\nabla \times \mathbf{E} = -\frac{\partial \mathbf{B}}{\partial t}$$

A (time)-varying electrical fields is always accompanied by a (time)-varying magnetic field

$$\nabla \times \mathbf{B} = \mu_0 \left( \mathbf{J} + \varepsilon_0 \frac{\partial \mathbf{E}}{\partial t} \right)$$

An electric current passing through a conductor is accompanied by a magnetic field circulating around that conductor

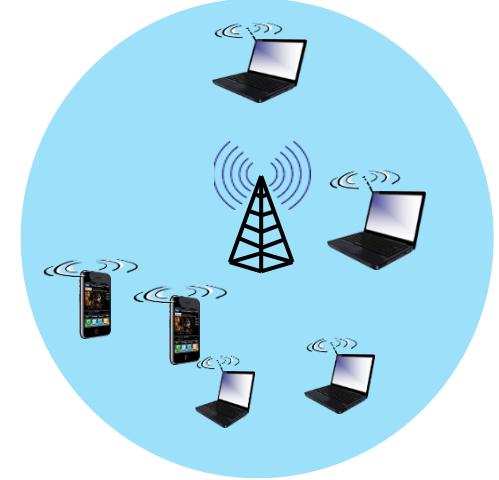
Without these equations, wireless communication would not be possible.  
They also allow us to identify certain challenges with radio communication...

# Wireless link characteristics (1)

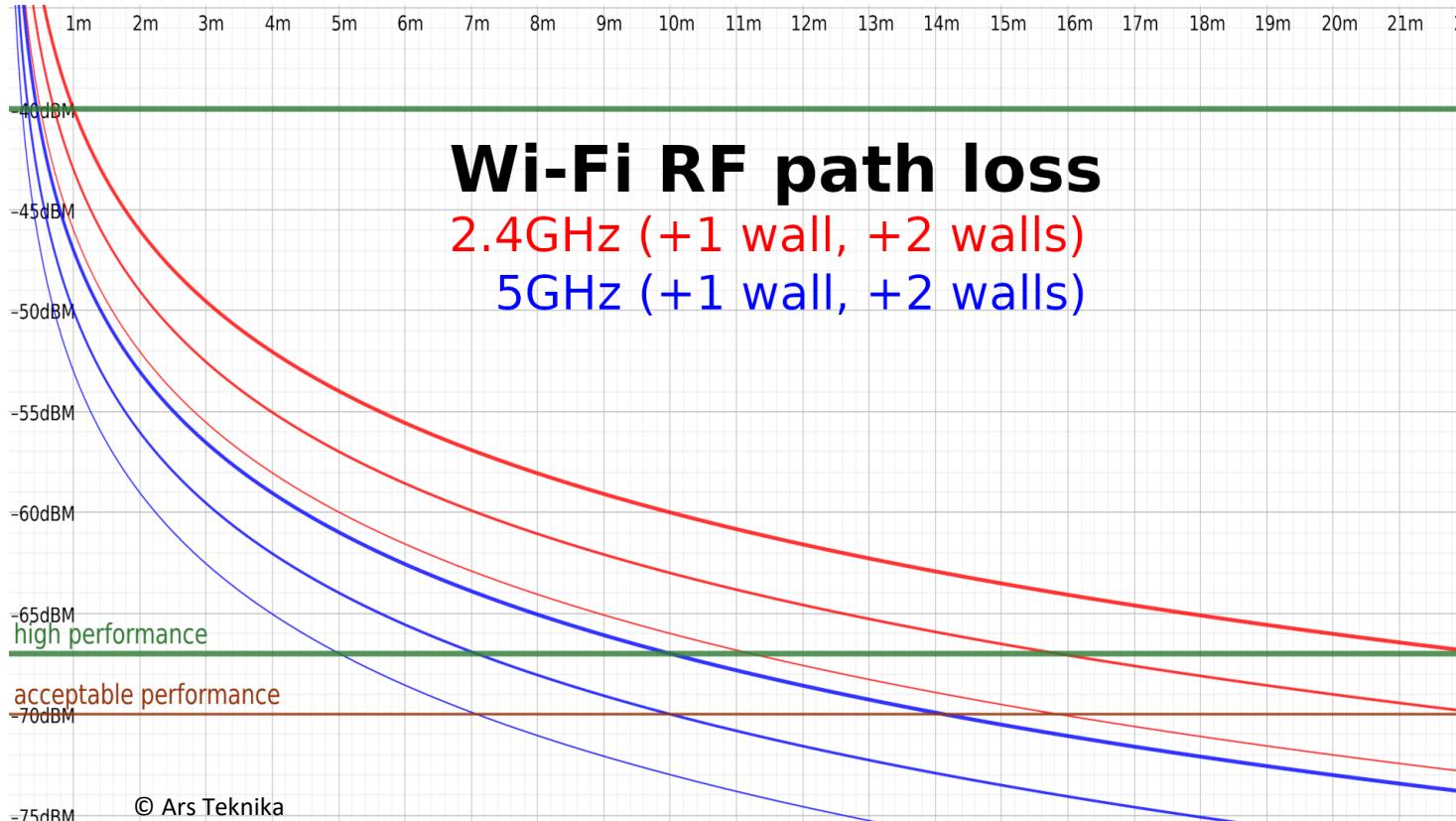
*important* differences from wired link ....

- **Decreased signal strength:** radio signal attenuates as it propagates through matter (path loss)
- **Interference from other sources:** wireless network frequencies (e.g., 2.4 GHz) shared by many devices (e.g., WiFi, cellular, motors): interference
- **Multipath propagation:** radio signal reflects off objects ground, arriving at destination at slightly different times

.... make communication across (even a point to point) wireless link much more “difficult”



# A little excursion to E-M field propagation



## Wi-Fi RF path loss

2.4GHz (+1 wall, +2 walls)  
5GHz (+1 wall, +2 walls)

- Path losses include propagation losses due to the “travel” of the radio wave in free space and losses due to diffraction and absorption.
- Path losses are frequency ( $\lambda = v/f$ ) and distance depending!
- $L_{path}(\lambda, d) \approx 20 \log_{10} \left( \frac{4\pi d}{\lambda} \right)$
- Lower wavelength  $\lambda$  (higher the frequency  $f$ ) cause higher path loss.
- More distance  $d$  also causes more path loss.
- Antenna design is very important (Gains)

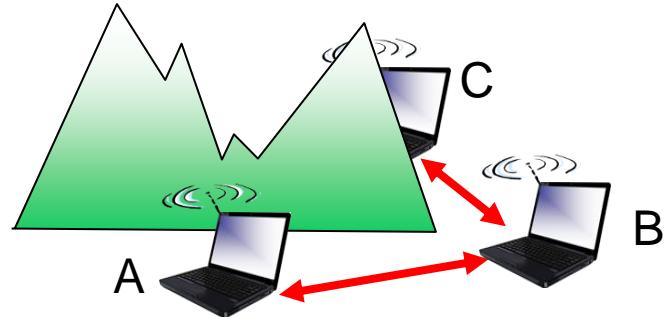
## Link Budget

$$P_{receiver} = P_{sender} + G_{sender} + G_{receiver} - L_{sender} - L_{receiver} - L_{path}(\lambda, d)$$

$P \Rightarrow$  Power,  $G \Rightarrow$  Gain,  $L_{sender/receiver} \Rightarrow$  Transmitter Losses,  $L_{path} \Rightarrow$  Path Loss

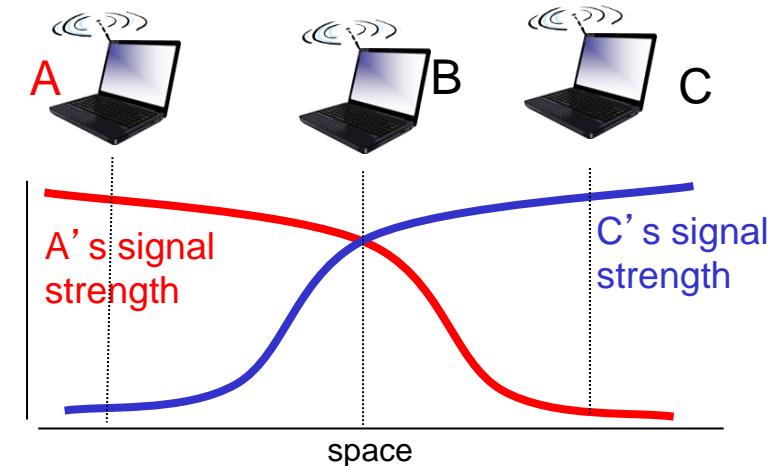
# Wireless link characteristics (2)

Multiple wireless senders, receivers create additional problems (beyond multiple access):



## Hidden terminal problem

- B, A hear each other
- B, C hear each other
- A, C can not hear each other means A, C unaware of their interference at B



## Signal attenuation:

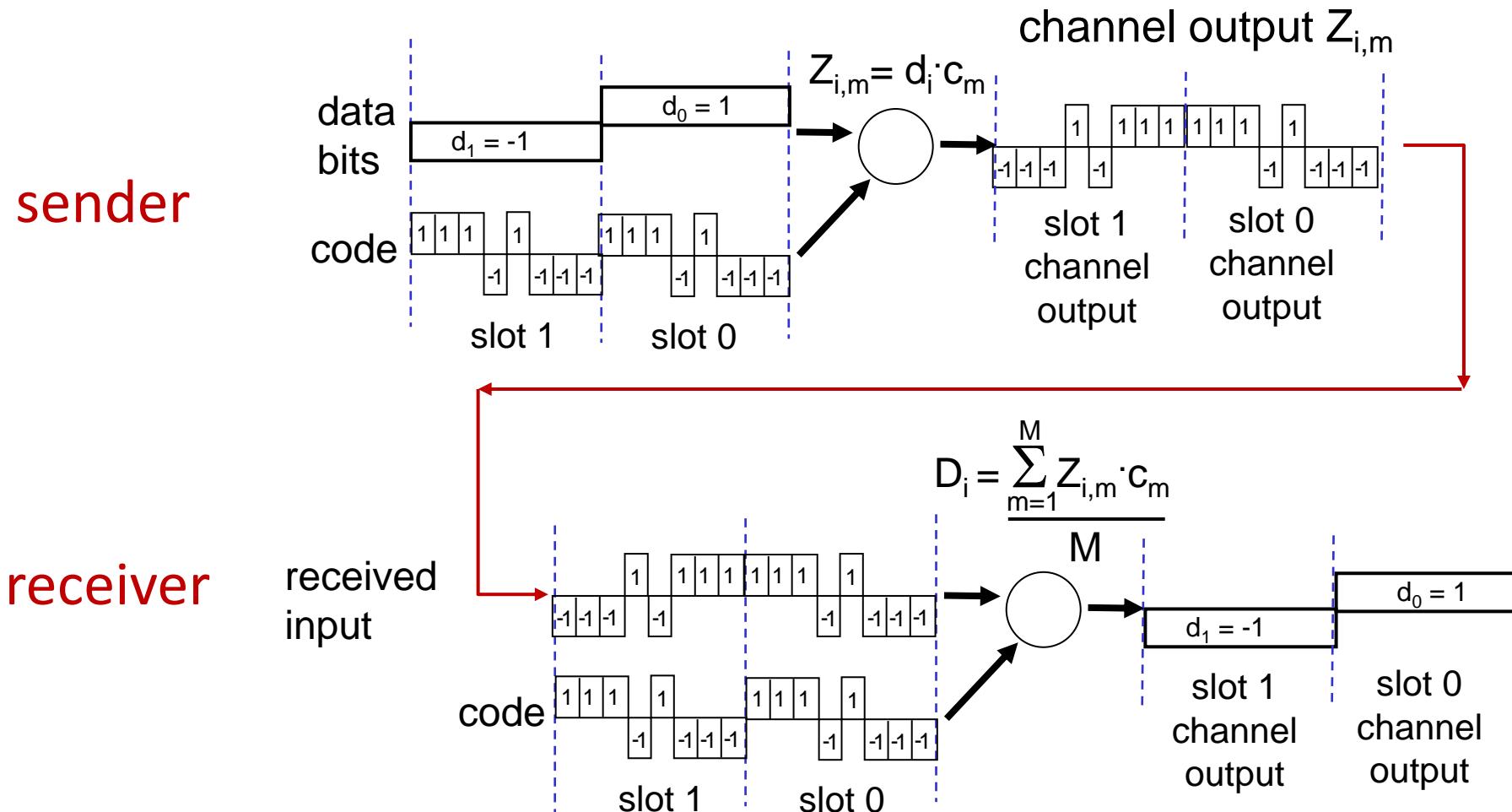
- B, A hear each other
- B, C hear each other
- A, C can not hear each other interfering at B

# Code Division Multiple Access (CDMA)

---

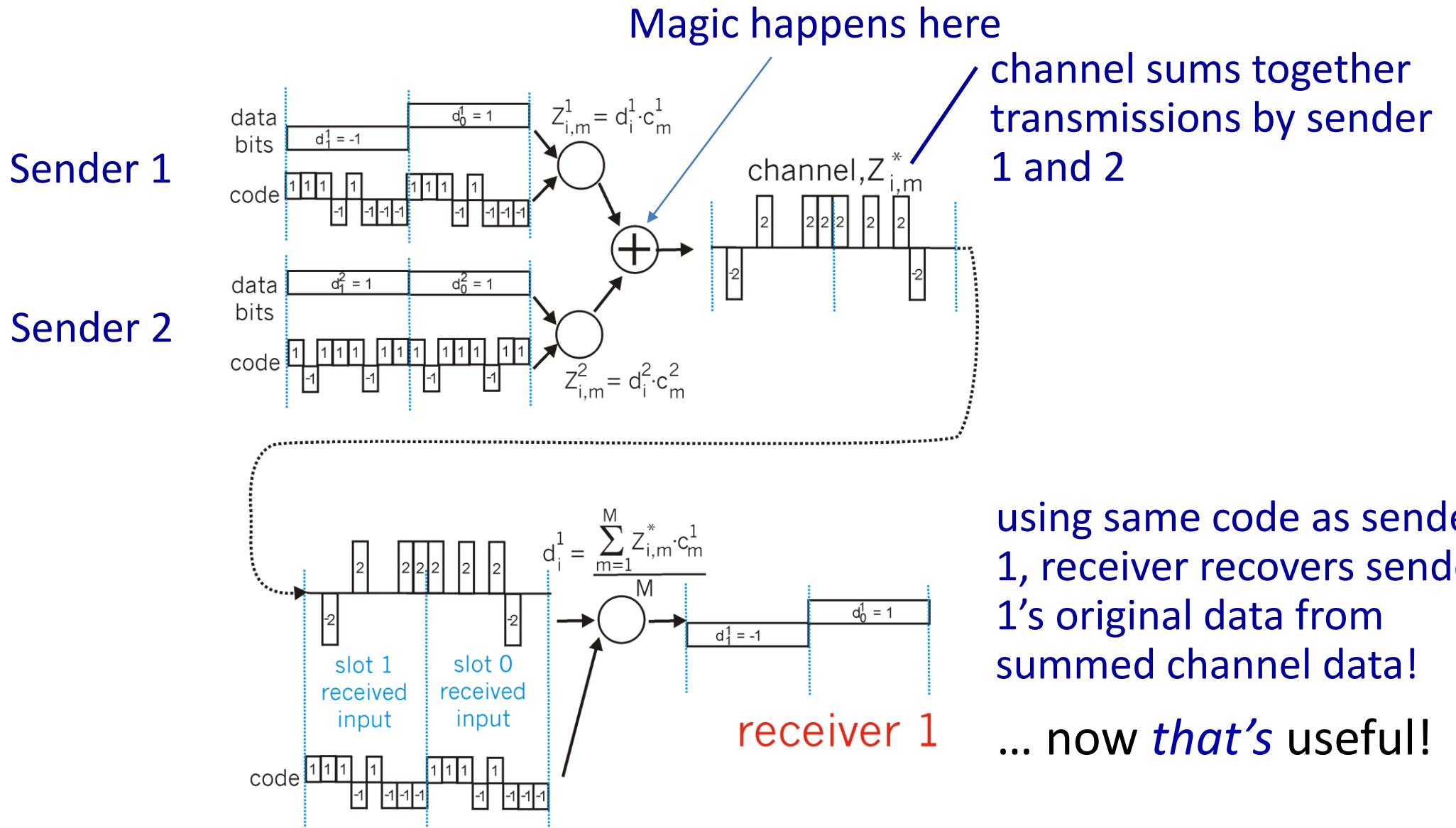
- CDMA is a channel access method such that **several transmitters can send information at the same time over a single communication channel.**
- Unique “code” assigned to each user; i.e., code set partitioning
  - All users share same frequency, but each user has own “chipping” sequence (i.e., code) to encode data.
  - Allows multiple users to “coexist” and transmit simultaneously with minimal interference (if codes are “orthogonal”).
- **Encoding:** inner product: (original data)  $\times$  (chipping sequence)
- **Decoding:** summed inner-product: (encoded data)  $\times$  (chipping sequence)

# CDMA encode/decode



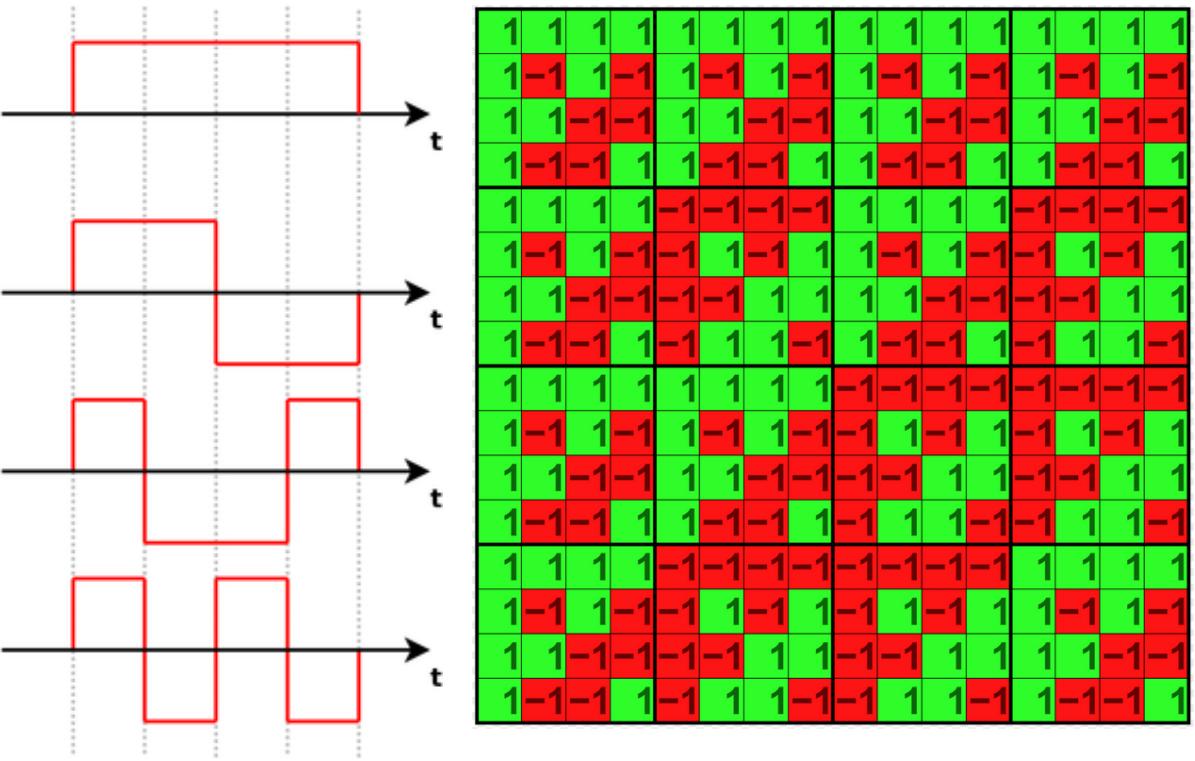
... but this is not really useful yet, is it?

# CDMA encode/decode magic



# Orthogonal codes

- **Encoding:** inner product: (original data)  $X$  (chipping sequence)
- **Decoding:** summed inner-product: (encoded data)  $X$  (chipping sequence)
- The chipping sequences must be orthogonal so that decoding can be successful for several clients.
  - Two elements  $u, v$  (of a vector space) with a bilinear form  $B$  are orthogonal when  $B(u, v) = 0$ .
    - In English: There is some function that linearly and for each argument separately “makes something zero”.
    - The dot product on  $\mathbb{R}^n$  is (conveniently) an example of a bilinear form

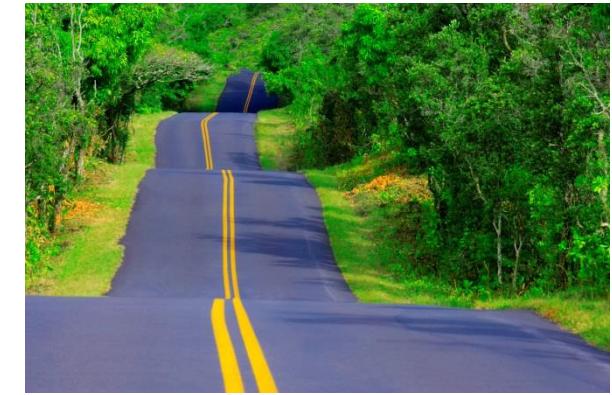


- The rows and columns of the **Welsh matrix** are orthogonal.
  - Their dot product is always zero.
- The (rows of the) Welsh matrix provide us with chipping sequences.

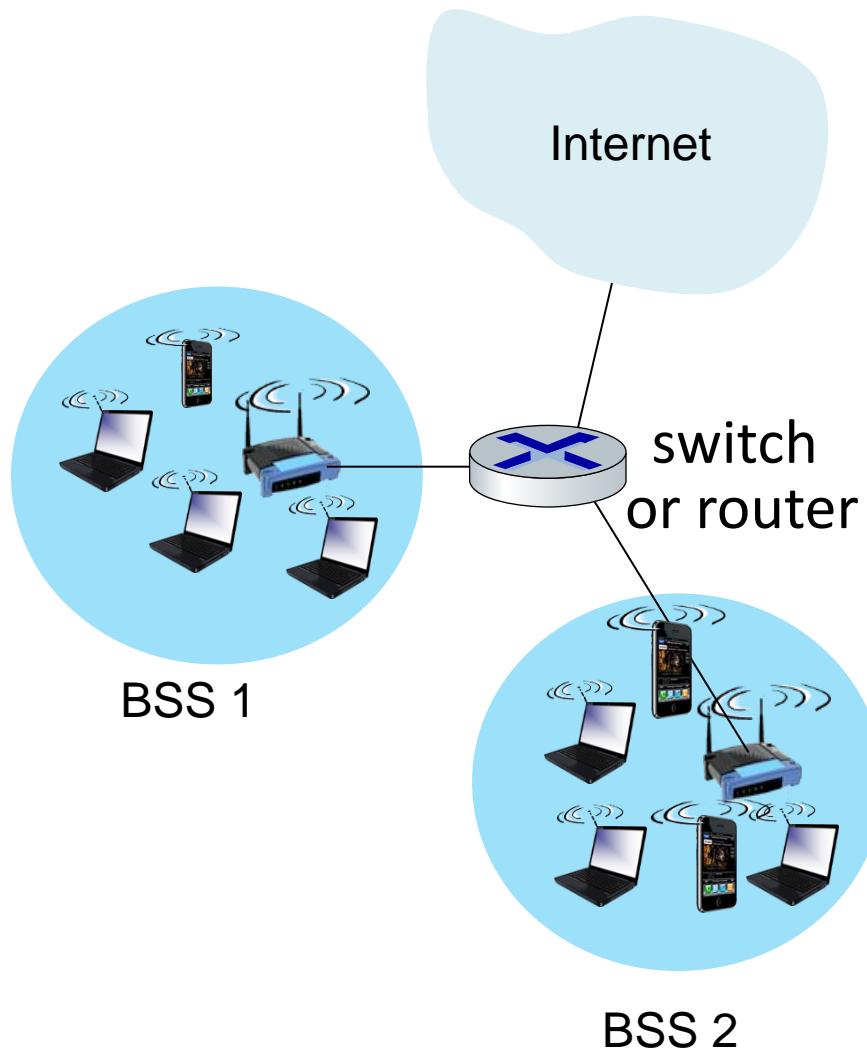
# Roadmap Data-Link Layer



- Introduction to Wireless Networks
- Wireless Links and Network Characteristics
- WiFi: 802.11 WLANs
- Cellular Networks: 4G and 5G
- Mobility management
  - Principles
  - Practice
  - Impact on higher-layer protocols



# 802.11 LAN architecture



- Wireless host communicates with base station  
**Base station = Access point (AP)**
- **Basic Service Set (BSS) (aka “cell”)** in infrastructure mode contains:
  - Wireless hosts
  - Access point (AP): base station
  - Ad hoc mode: hosts only

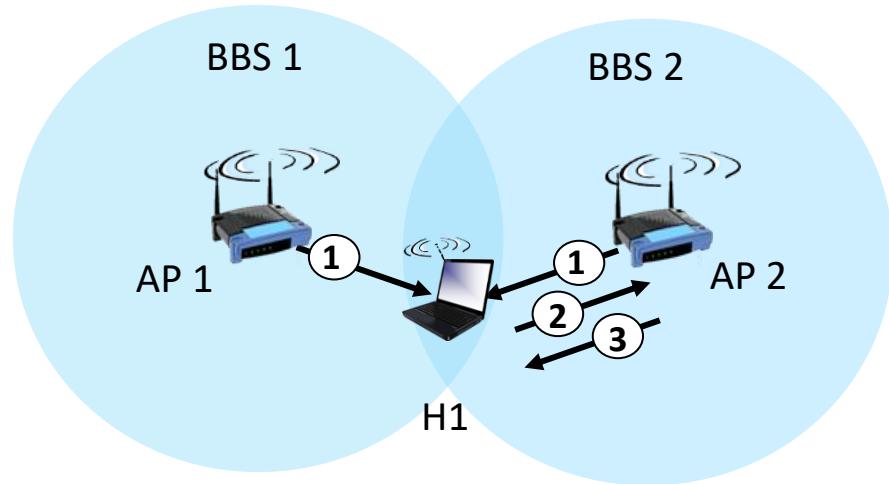
# 802.11: Channels, association

---

- Spectrum divided into channels at different frequencies
  - AP admin chooses frequency for AP
  - interference possible: channel can be same as that chosen by neighboring AP!
- Arriving host: must **associate** with an AP
  - scans channels, listening for *beacon frames* containing AP's name (SSID) and MAC address
  - selects AP to associate with
  - then may perform authentication [Chapter 8]
  - then typically run DHCP to get IP address in AP's subnet

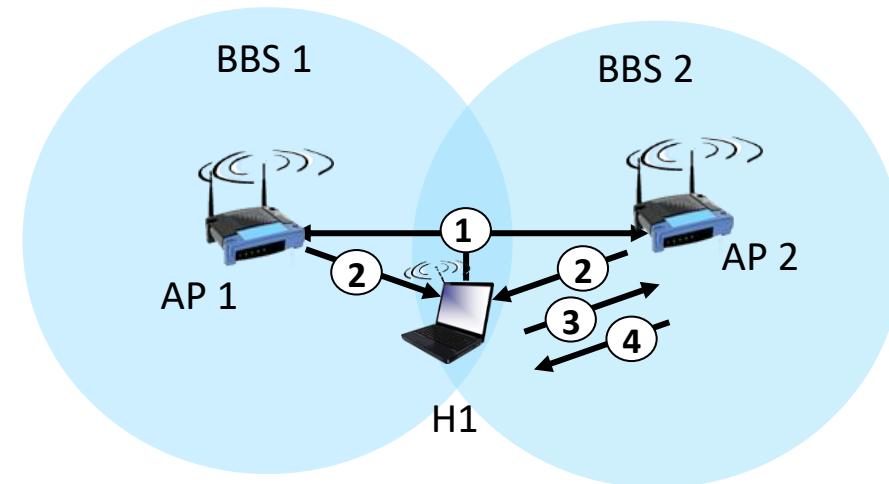


# 802.11: passive/active scanning



## Passive scanning:

- (1) beacon frames sent from APs
- (2) association Request frame sent: H1 to selected AP
- (3) association Response frame sent from selected AP to H1

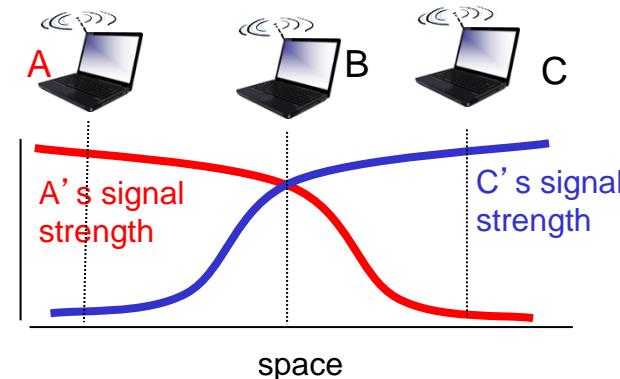
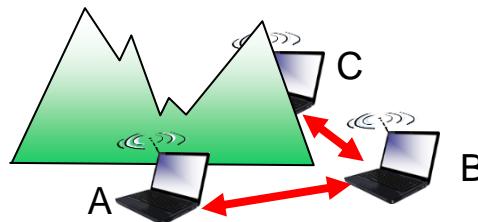


## Active scanning:

- (1) Probe Request frame broadcast from H1
- (2) Probe Response frames sent from APs
- (3) Association Request frame sent: H1 to selected AP
- (4) Association Response frame sent from selected AP to H1

# IEEE 802.11: multiple access

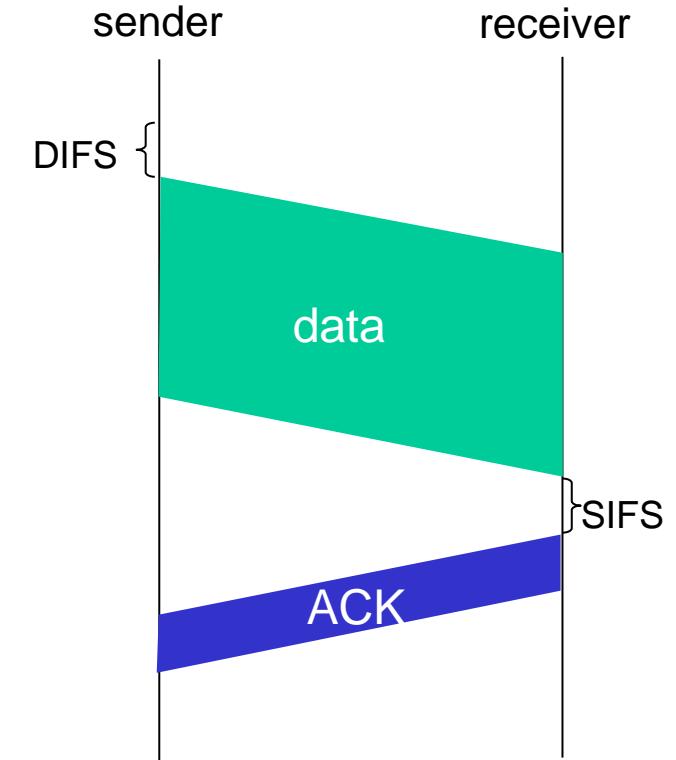
- Avoid collisions:  $2^+$  nodes transmitting at same time
- 802.11: CSMA - sense before transmitting
  - Don't collide with detected ongoing transmission by another node
- 802.11: *no collision detection!*
  - Difficult to sense collisions: high transmitting signal, weak received signal due to fading
  - Cannot sense all collisions in any case: hidden terminal, fading
  - Goal: *avoid collisions*: CSMA/CollisionAvoidance



# IEEE 802.11 MAC Protocol: CSMA/CA

## 802.11 sender

- 1 If sense channel idle for **DIFS** then  
transmit entire frame (no CD)
- 2 If sense channel busy then  
start random backoff time  
timer counts down while channel idle  
transmit when timer expires  
if no ACK, increase random backoff interval, repeat 2



## 802.11 receiver

If frame received OK  
return ACK after **SIFS** (ACK needed due to hidden terminal problem)

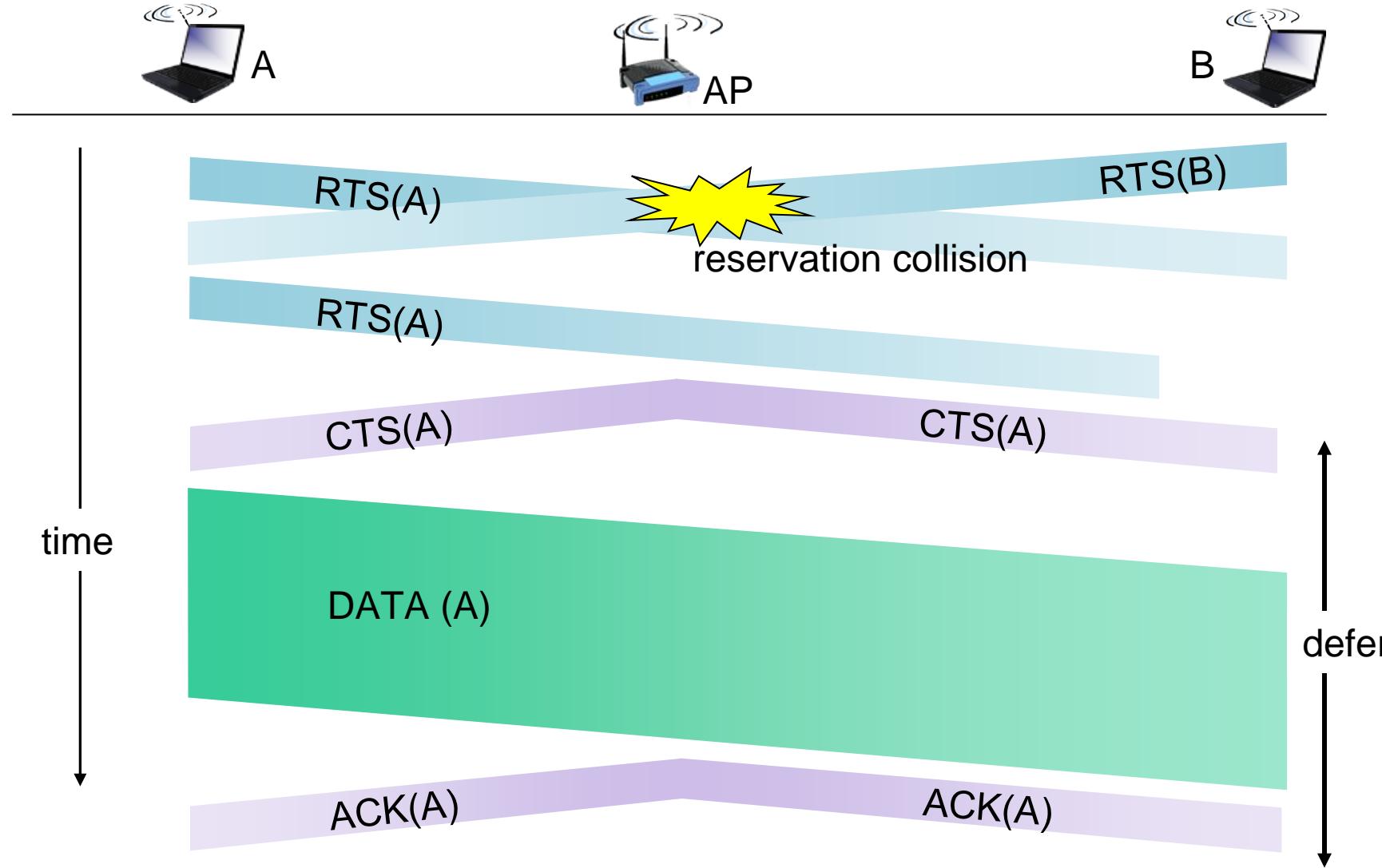
# Avoiding collisions (more)

---

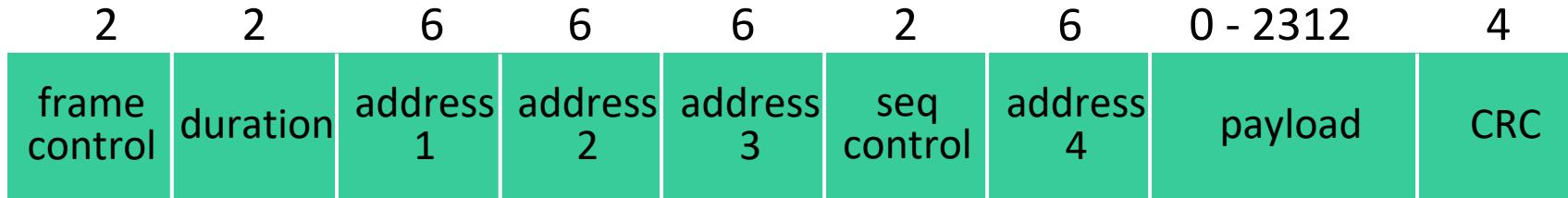
Idea: sender “reserves” channel use for data frames using small reservation packets

- Sender first transmits *small request-to-send (RTS)* packet to BS using CSMA
  - RTSs may still collide with each other (but they’re short)
- BS broadcasts *clear-to-send CTS* in response to RTS
- CTS heard by all nodes
  - sender transmits data frame
  - other stations defer transmissions

# Collision Avoidance: RTS-CTS exchange



# 802.11 frame: Addressing



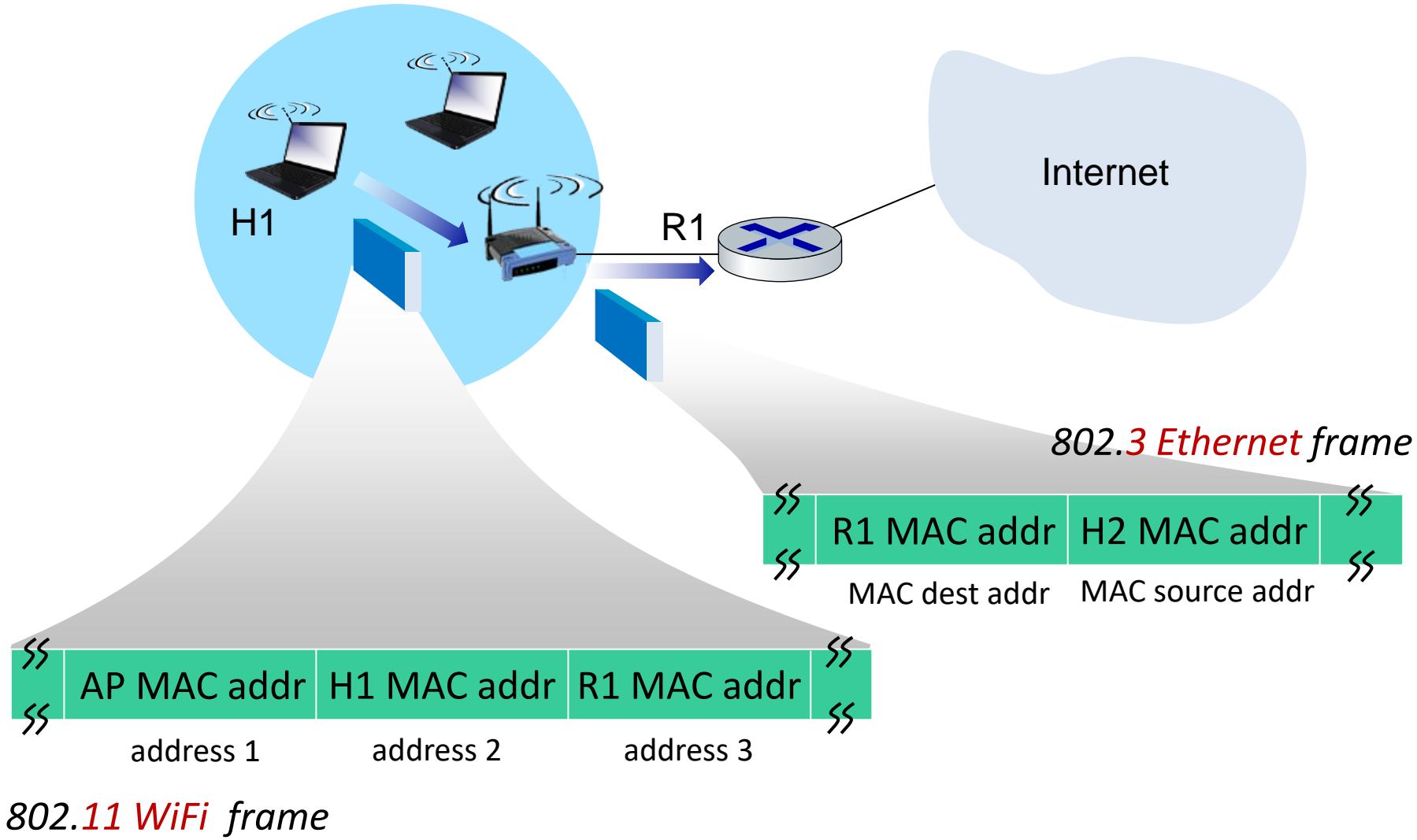
**Address 1:** MAC address of wireless host or AP to receive this frame

**Address 2:** MAC address of wireless host or AP transmitting this frame

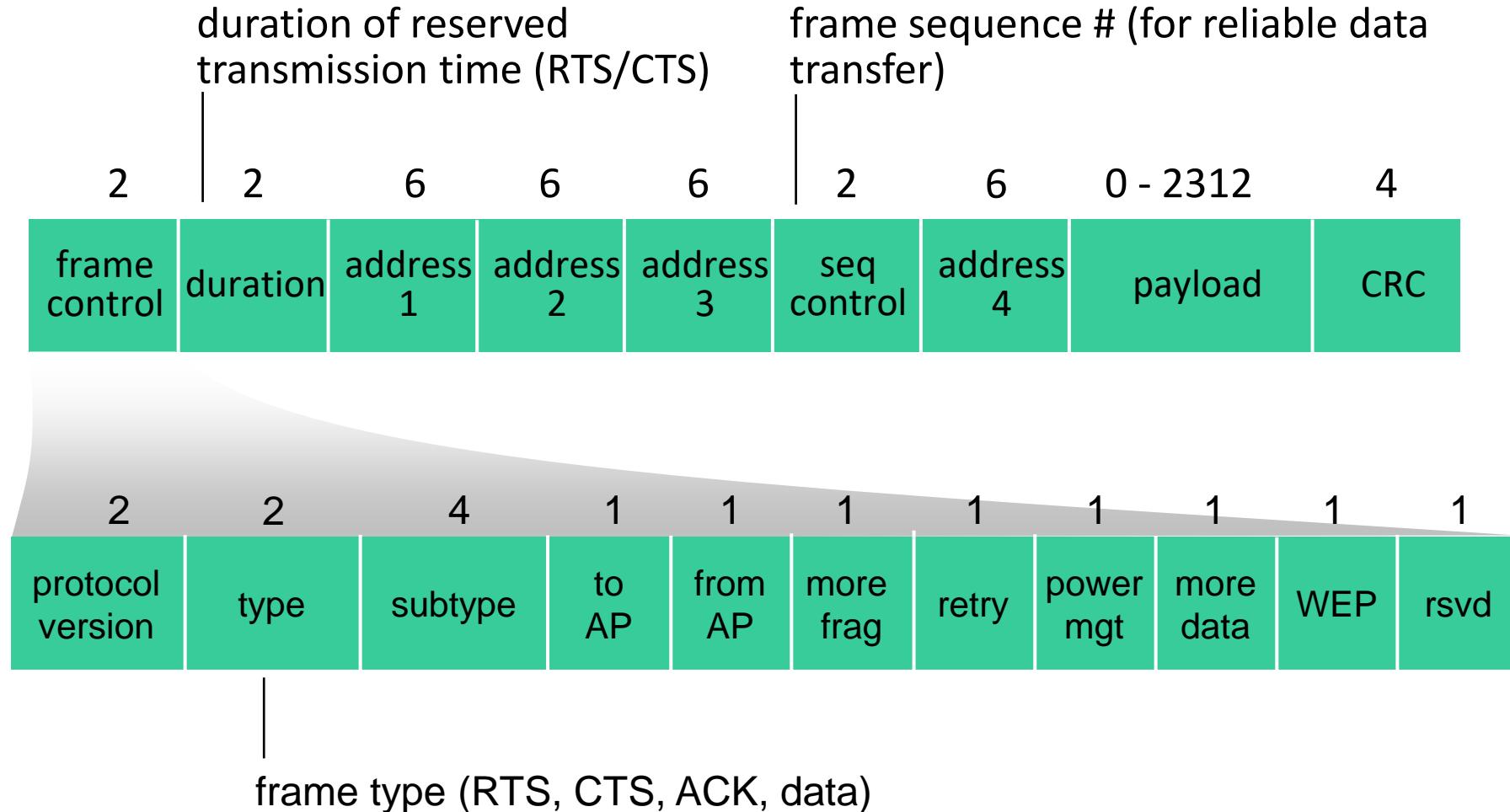
**Address 4:** used only in ad hoc mode

**Address 3:** MAC address of router interface to which AP is attached

# 802.11 frame: Addressing

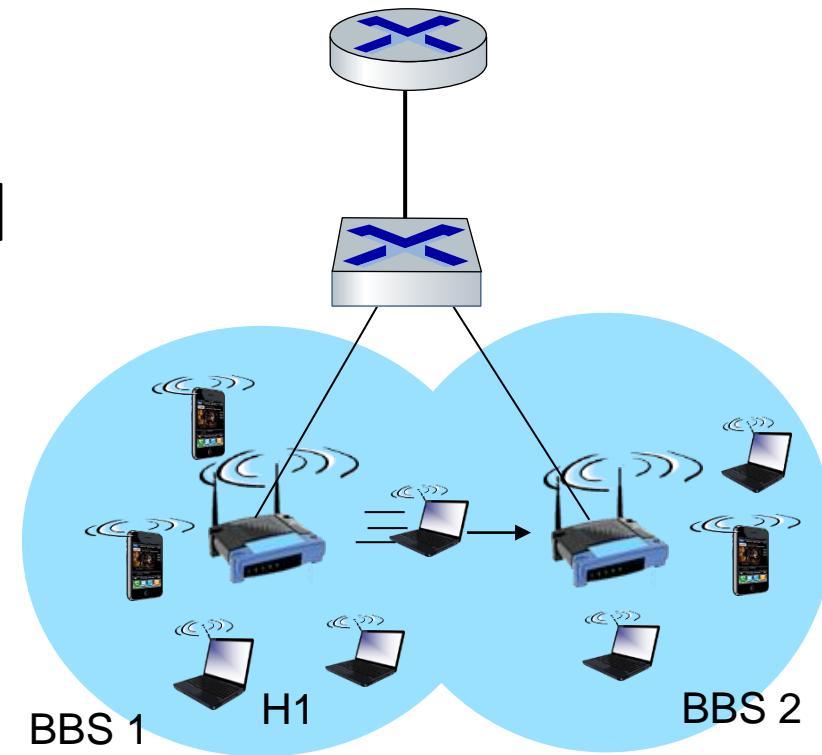


# 802.11 frame: Addressing



# 802.11: Mobility within same subnet

- H1 remains in same IP subnet:  
**IP address can remain same**
- Switch: Which AP is associated with H1?
  - self-learning (Ch. 6): switch will see frame from H1 and “remember” which switch port can be used to reach H1



# 802.11: Advanced capabilities

---

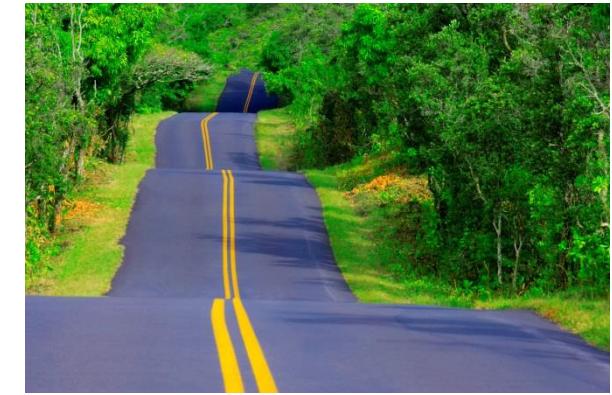
## Power management

- Node-to-AP: “I am going to sleep until next beacon frame”
  - AP knows not to transmit frames to this node
  - Node wakes up before next beacon frame
- Beacon frame: contains list of mobiles with AP-to-mobile frames waiting to be sent
  - Node will stay awake if AP-to-mobile frames to be sent; otherwise sleep again until next beacon frame

# Roadmap Data-Link Layer



- Introduction to Wireless Networks
- Wireless Links and Network Characteristics
- WiFi: 802.11 WLANs
- Cellular Networks: 4G and 5G
- Mobility management
  - Principles
  - Practice
  - Impact on higher-layer protocols



# 4G/5G cellular networks

---

- *The solution for wide-area mobile Internet*
- Widespread deployment/use:
  - More mobile-broadband-connected devices than fixed-broadband-connected devices (5-1 in 2019)!
  - 4G availability: 97% of time in Korea (90%+ in US, >98% in Sweden)
- Transmission rates up to 100's Mbps
- Technical standards: 3rd Generation Partnership Project (3GPP)
  - [www.3gpp.org](http://www.3gpp.org)
  - 4G: Long-Term Evolution (LTE) standard

# 4G/5G cellular networks

---

## *similarities to wired Internet*

- Edge/core distinction, but both belong to same carrier
- Global cellular network: a network of networks
- Widespread use of protocols we've studied: HTTP, DNS, TCP, UDP, IP, NAT, separation of data/control planes, SDN, Ethernet
- Interconnected to wired Internet

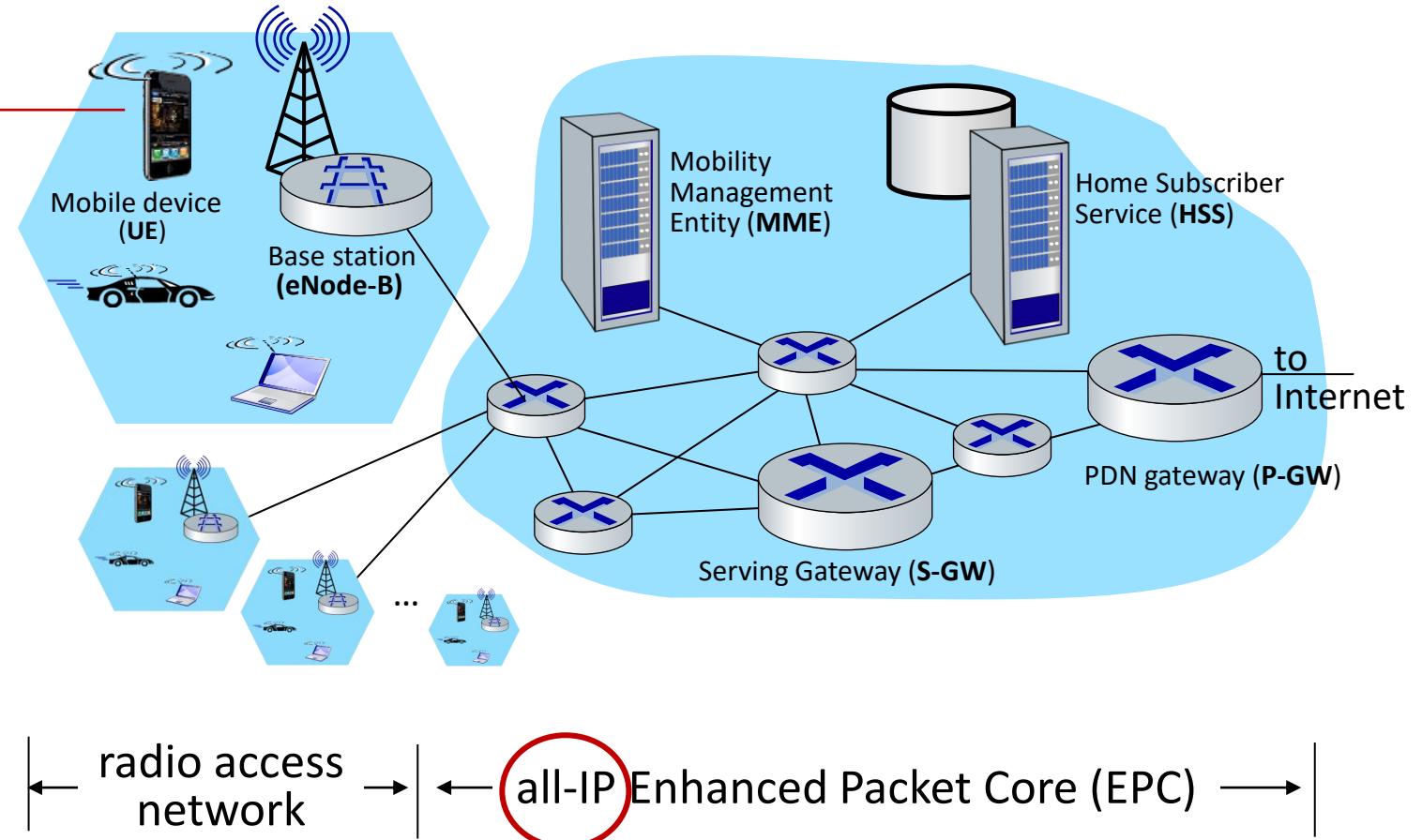
## *differences from wired Internet*

- Different wireless link layer
- Focus on mobility
- User “identity” (via SIM card)
- Business model: users subscribe to a cellular provider
  - Strong notion of “home network” versus roaming on visited nets
  - Global access, with authentication infrastructure, and inter-carrier settlements

# Elements of 4G LTE architecture

## Mobile device:

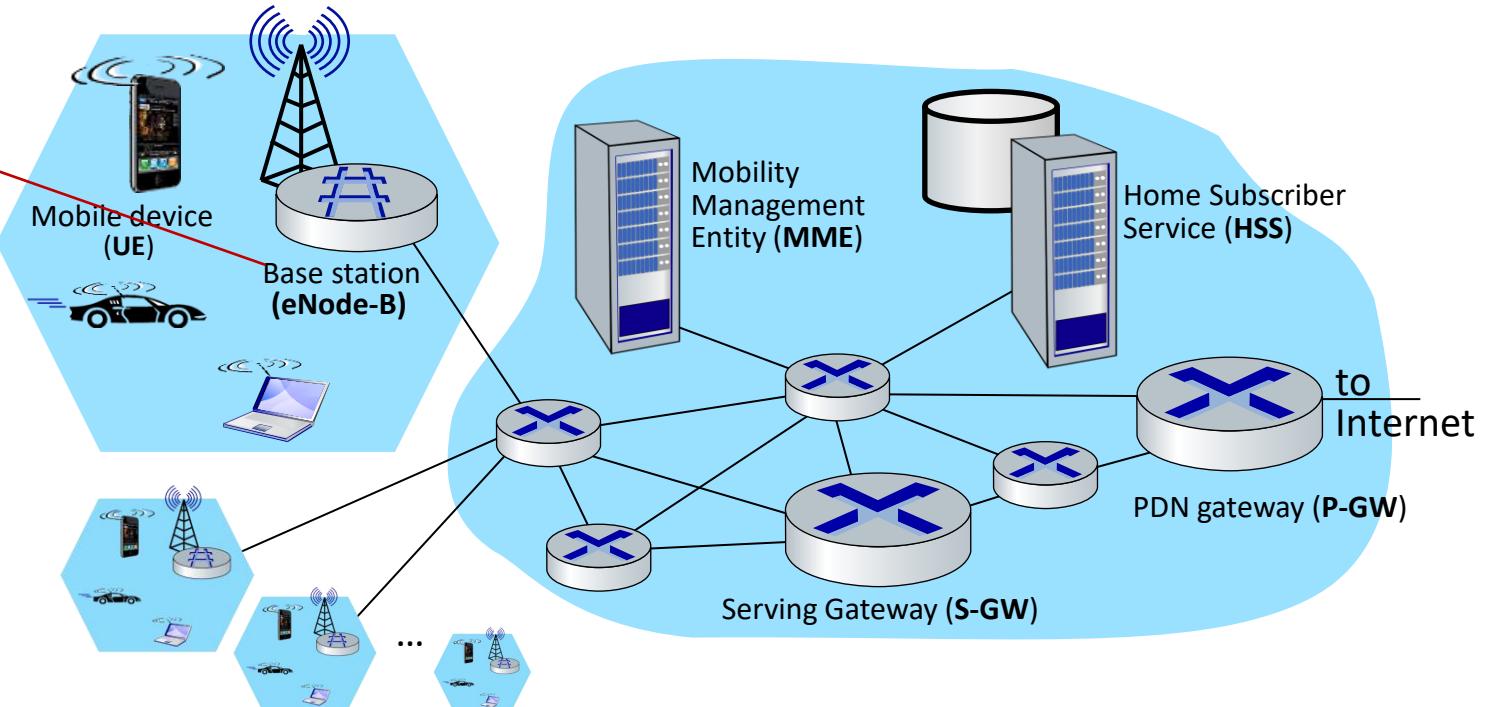
- Smartphone, tablet, laptop, IoT, ... with 4G LTE radio
- 64-bit International Mobile Subscriber Identity (IMSI), stored on SIM (Subscriber Identity Module) card
- LTE jargon: User Equipment (UE)



# Elements of 4G LTE architecture

## Base station:

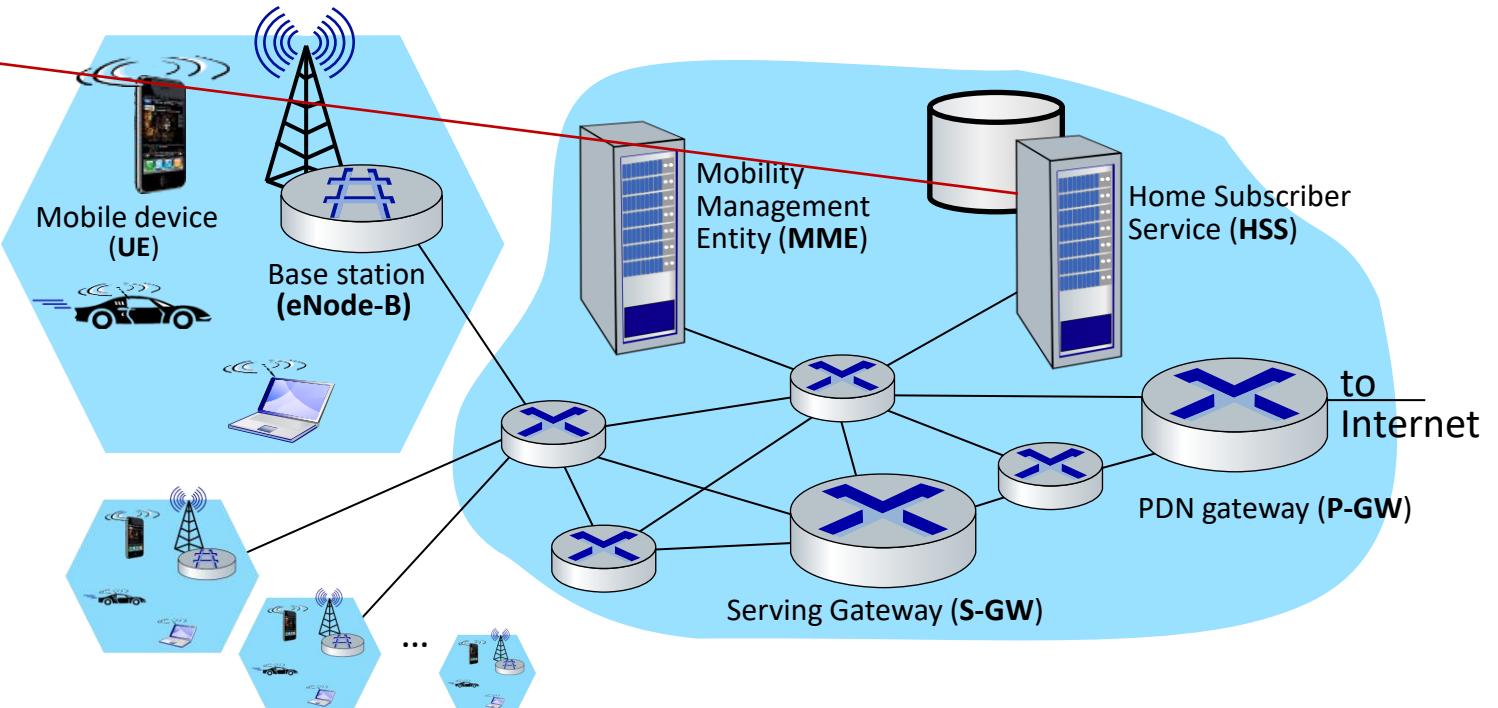
- At “edge” of carrier’s network
- Manages wireless radio resources, mobile devices in its coverage area (“cell”)
- Coordinates device authentication with other elements
- Similar to WiFi AP but:
  - active role in user mobility
  - coordinates with nearby base stations to optimize radio use
- LTE jargon: eNode-B



# Elements of 4G LTE architecture

## Home Subscriber Service

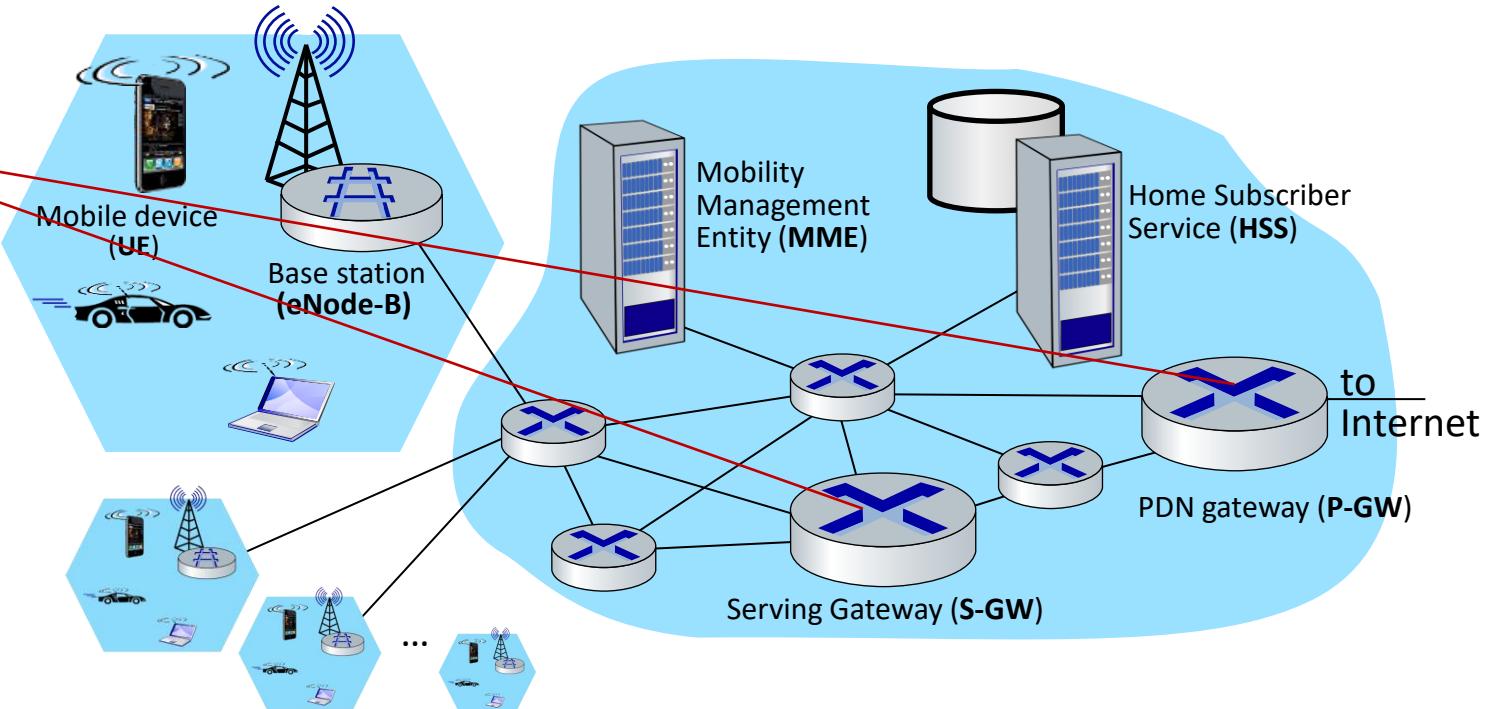
- Stores info about mobile devices for which the HSS's network is their "home network"
- Works with MME in device authentication



# Elements of 4G LTE architecture

## Serving Gateway (S-GW), PDN Gateway (P-GW)

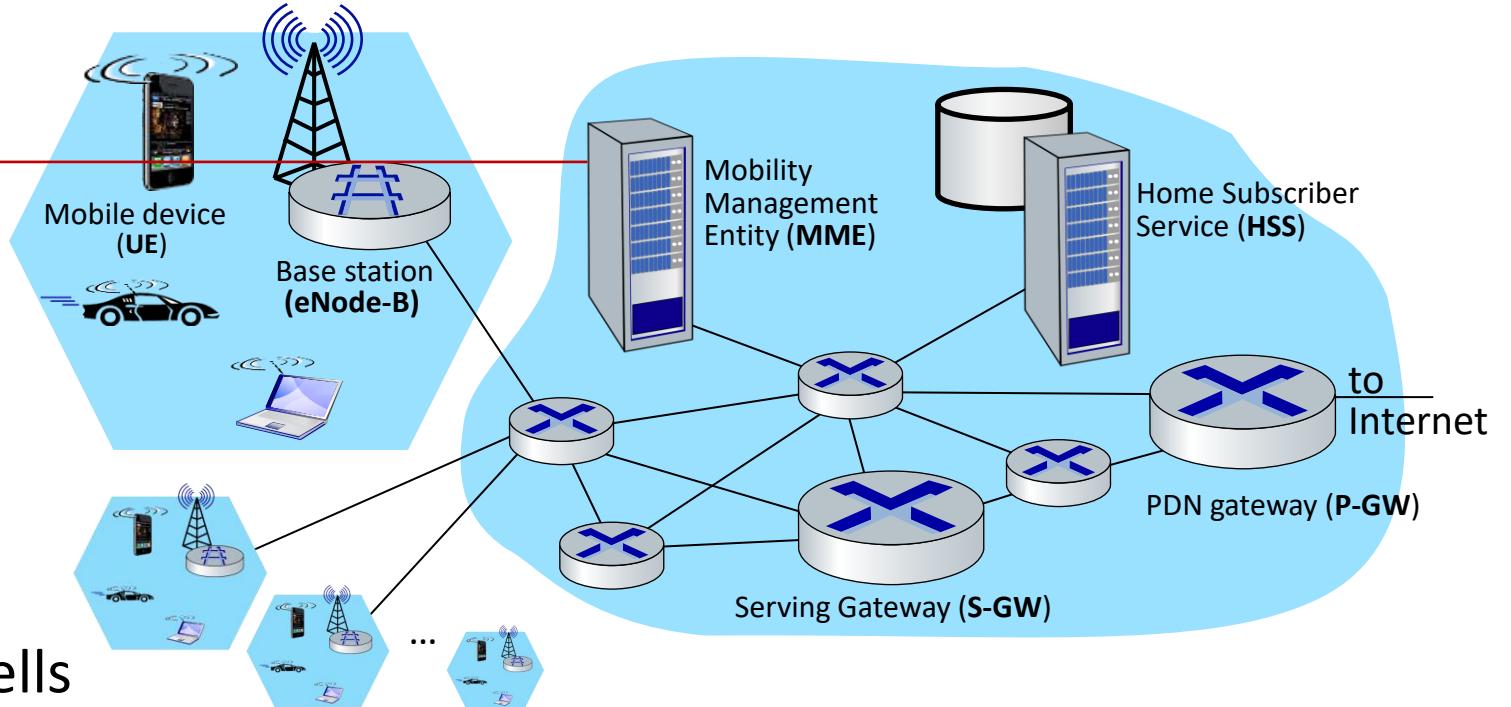
- Lie on data path from mobile to/from Internet
- P-GW
  - gateway to mobile cellular network
  - Looks like any other internet gateway router
  - provides NAT services



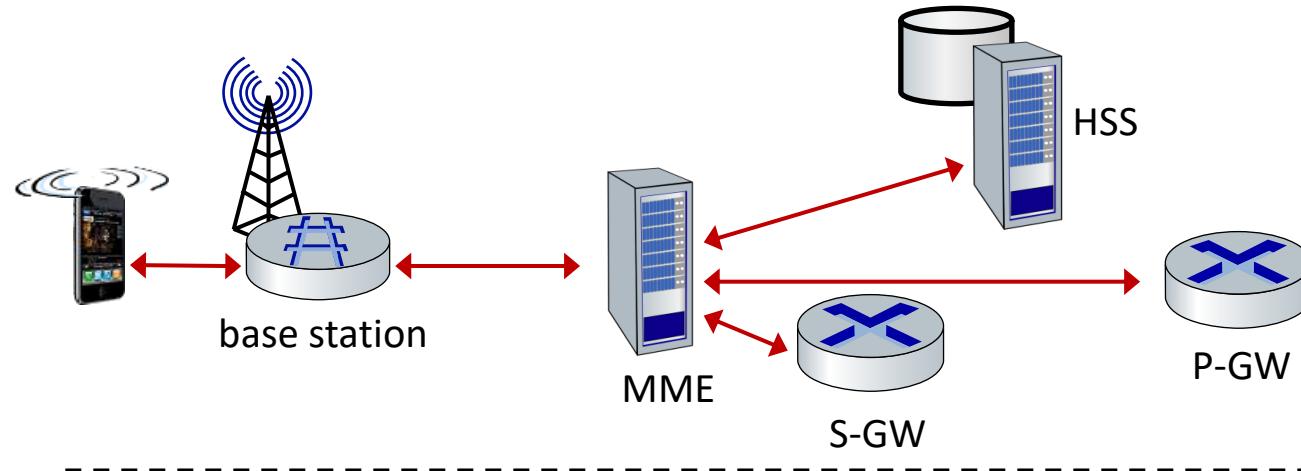
# Elements of 4G LTE architecture

## Mobility Management Entity

- Device authentication (device-to-network, network-to-device) coordinated with mobile home network HSS
- Mobile device management:
  - Device handover between cells
  - Tracking/paging device location
- Path (tunneling) setup from mobile device to P-GW

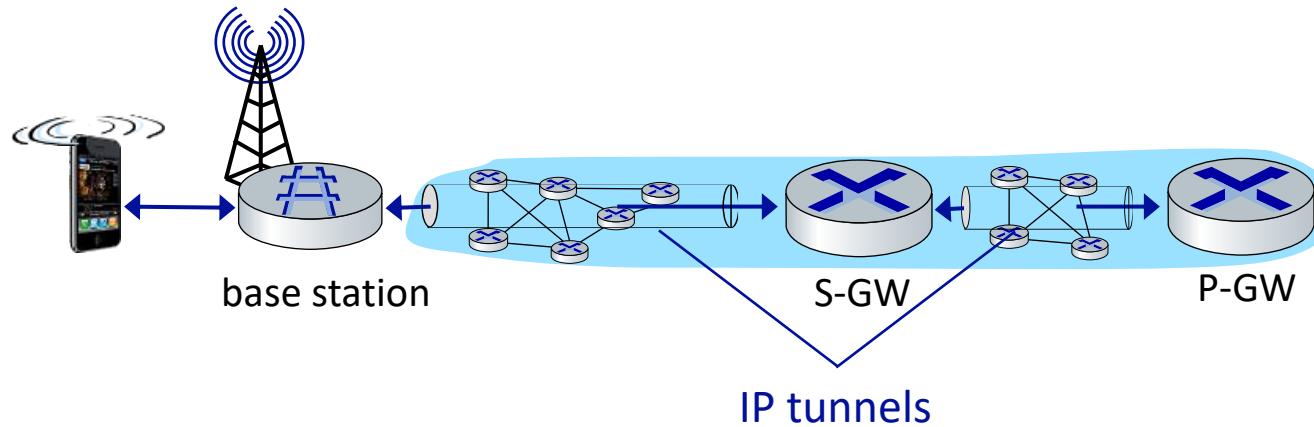


# LTE: Data plane control plane separation



## control plane

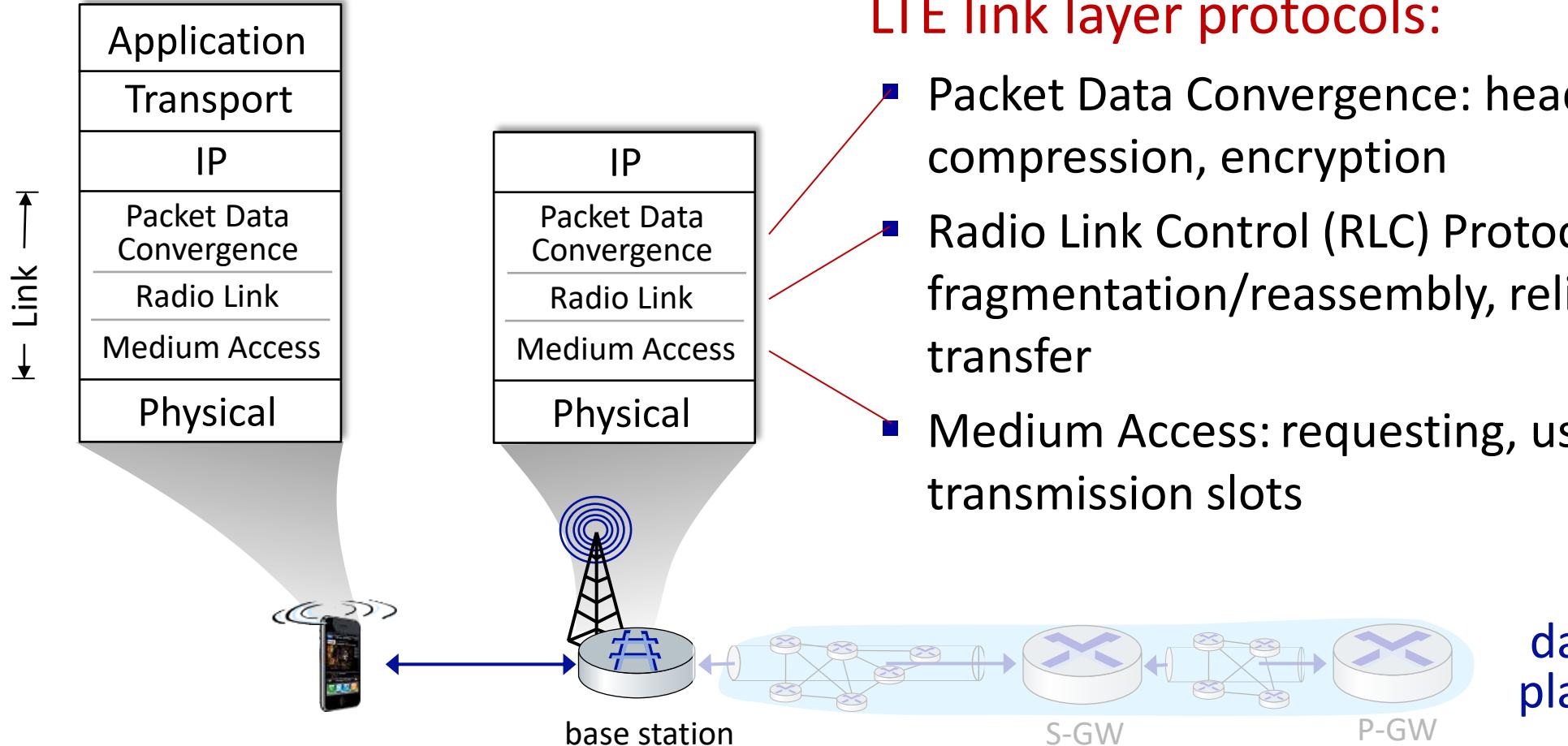
- new protocols for mobility management , security, authentication (later)



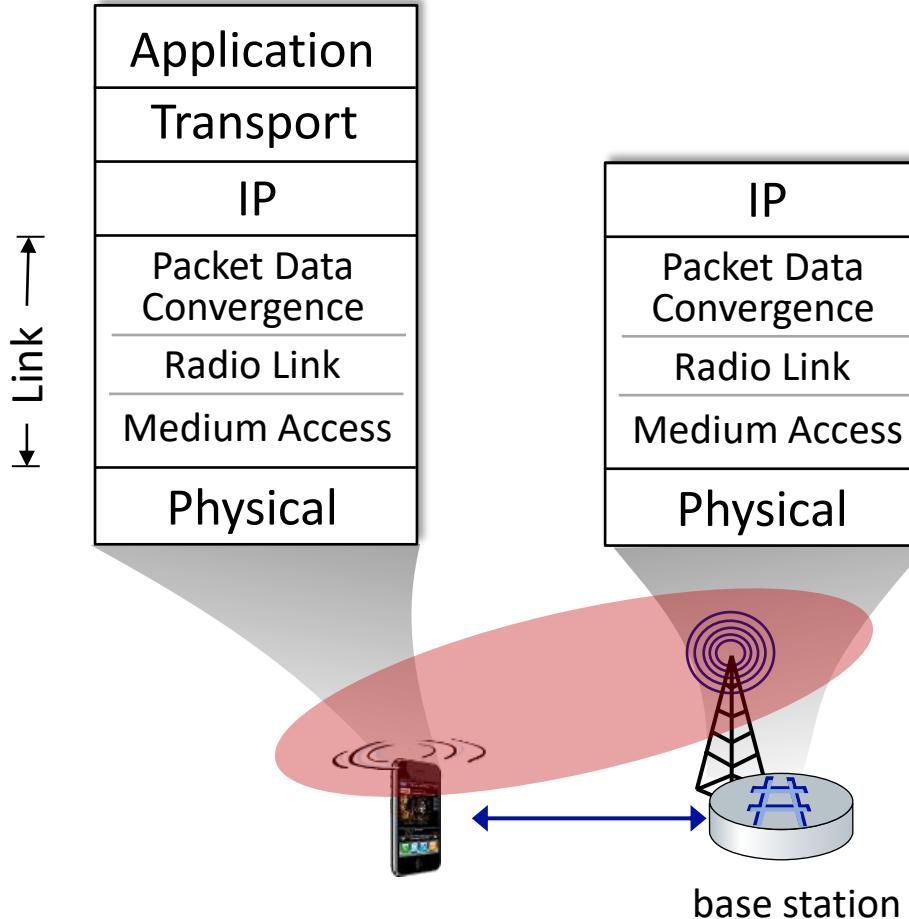
## data plane

- new protocols at link, physical layers
- extensive use of tunneling to facilitate mobility

# LTE data plane protocol stack: first hop



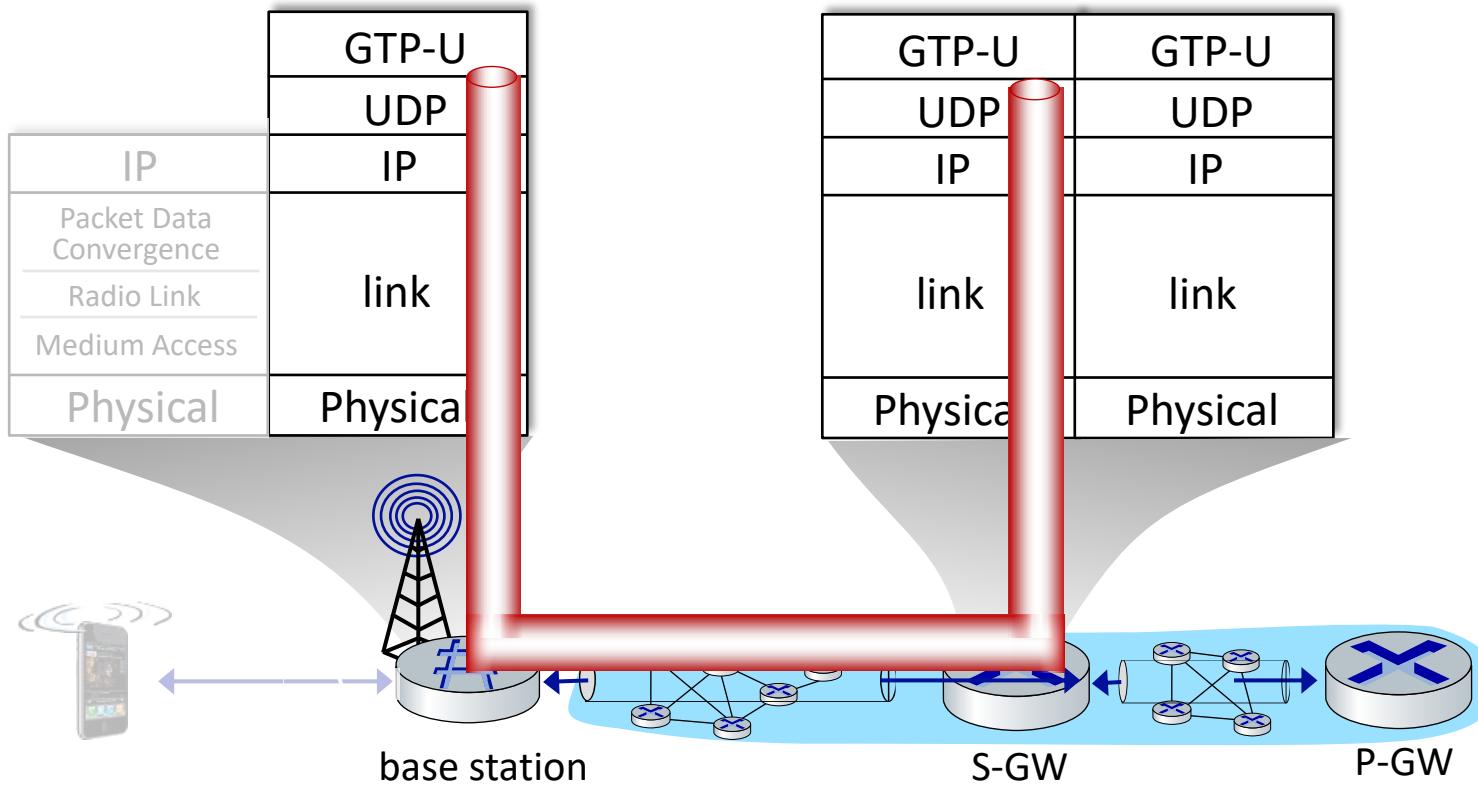
# LTE data plane protocol stack: First hop



## LTE radio access network:

- **downstream channel:** FDM, TDM within frequency channel (OFDM - orthogonal frequency division multiplexing)
  - “orthogonal”: minimal interference between channels
- **upstream:** FDM, TDM similar to OFDM
- each active mobile device allocated two or more 0.5 ms time slots over 12 frequencies
  - scheduling algorithm not standardized – up to operator
  - 100's Mbps per device possible

# LTE data plane protocol stack: Packet core



## tunneling:

- mobile datagram encapsulated using GPRS Tunneling Protocol (GTP), sent inside UDP datagram to S-GW
- S-GW re-tunnels datagrams to P-GW
- supporting mobility: only tunneling endpoints change when mobile user moves

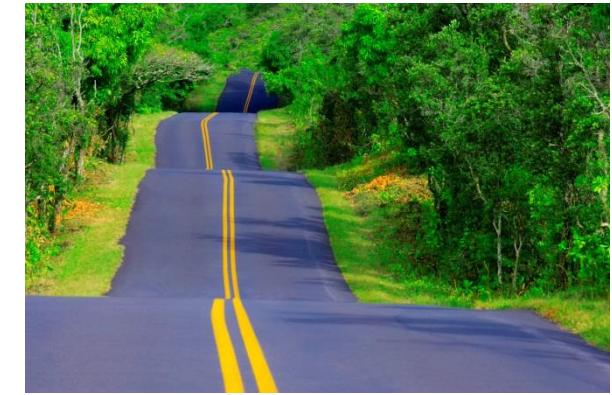
# On to 5G!

---

- **Goal:** 10x increase in peak bitrate, 10x decrease in latency, 100x increase in traffic capacity over 4G
- **5G NR (new radio):**
  - Two frequency bands: FR1 (450 MHz–6 GHz) and FR2 (24 GHz–52 GHz): millimeter wave frequencies
  - Not backwards-compatible with 4G
  - MIMO: multiple directional antennae
- **Millimeter wave frequencies:** much higher data rates, but over shorter distances (remember signal-loss depends on wavelength!).
  - Pico-cells: cells diameters: 10-100 m
  - Massive, dense deployment of new base stations required

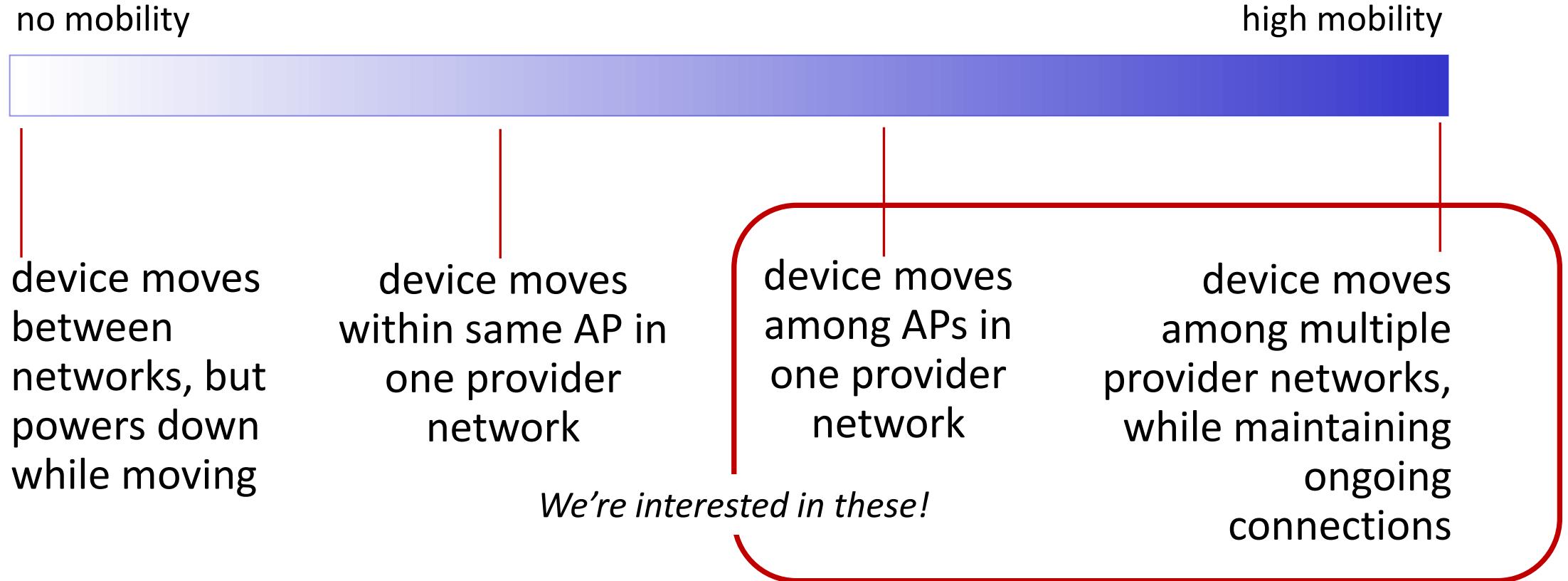
# Roadmap Data-Link Layer

- Introduction to Wireless Networks
- Wireless Links and Network Characteristics
- WiFi: 802.11 WLANs
- Cellular Networks: 4G and 5G
- **Mobility management**
  - Principles
  - Practice
  - Impact on higher-layer protocols



# What is mobility?

- Spectrum of mobility, from the **network** perspective:



# Mobility approaches

---

- let network (routers) handle it:
  - routers advertise well-known name, address (e.g., permanent 32-bit IP address), or number (e.g., cell #) of visiting mobile node via usual routing table exchange
  - Internet routing could do this already *with no* changes! Routing tables indicate where each mobile located via longest prefix match!

# Mobility approaches

---

- let network (routers) handle it:
  - routers advertise well-known address (e.g., permanent 32-bit IP address), or number of visiting mobile node via usual routing table exchange
  - Internet routing could do the same *ready with no changes!* Routing tables indicate where each mobile located via longest prefix match!
- let end-systems handle it: functionality at the “edge”
  - *indirect routing*: communication from correspondent to mobile goes through home network, then forwarded to remote mobile
  - *direct routing*: correspondent gets foreign address of mobile, send directly to mobile

# Contacting a mobile friend:

Consider friend frequently changing locations, how do you find him/her?

- search all phone books?
- expect her to let you know where he/she is?
- call their parents?
- Facebook!

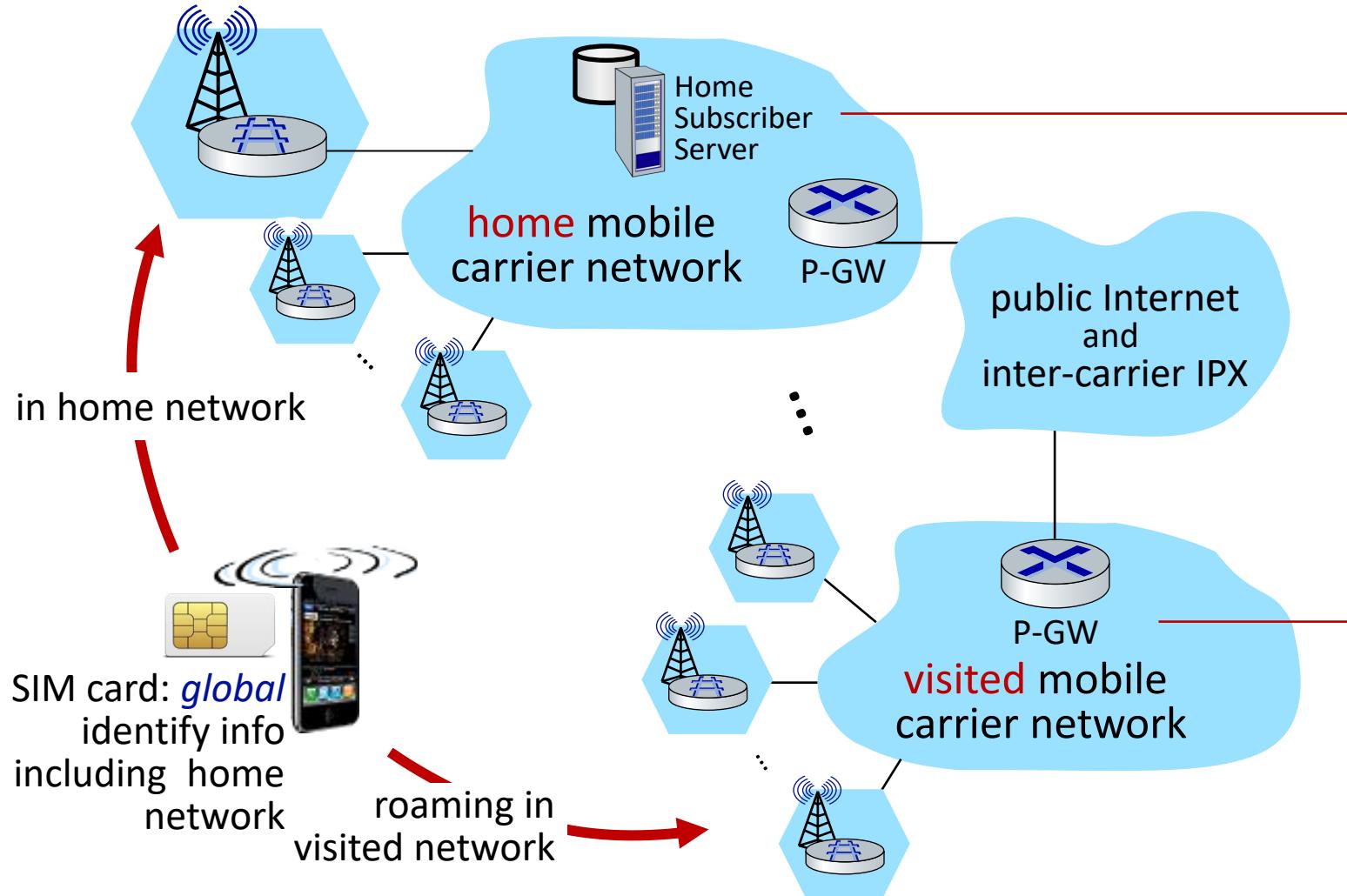
I wonder where Alice moved to?



The importance of having a “home”:

- a definitive source of information about you
- a place where people can find out where you are

# Home network, visited network: 4G/5G



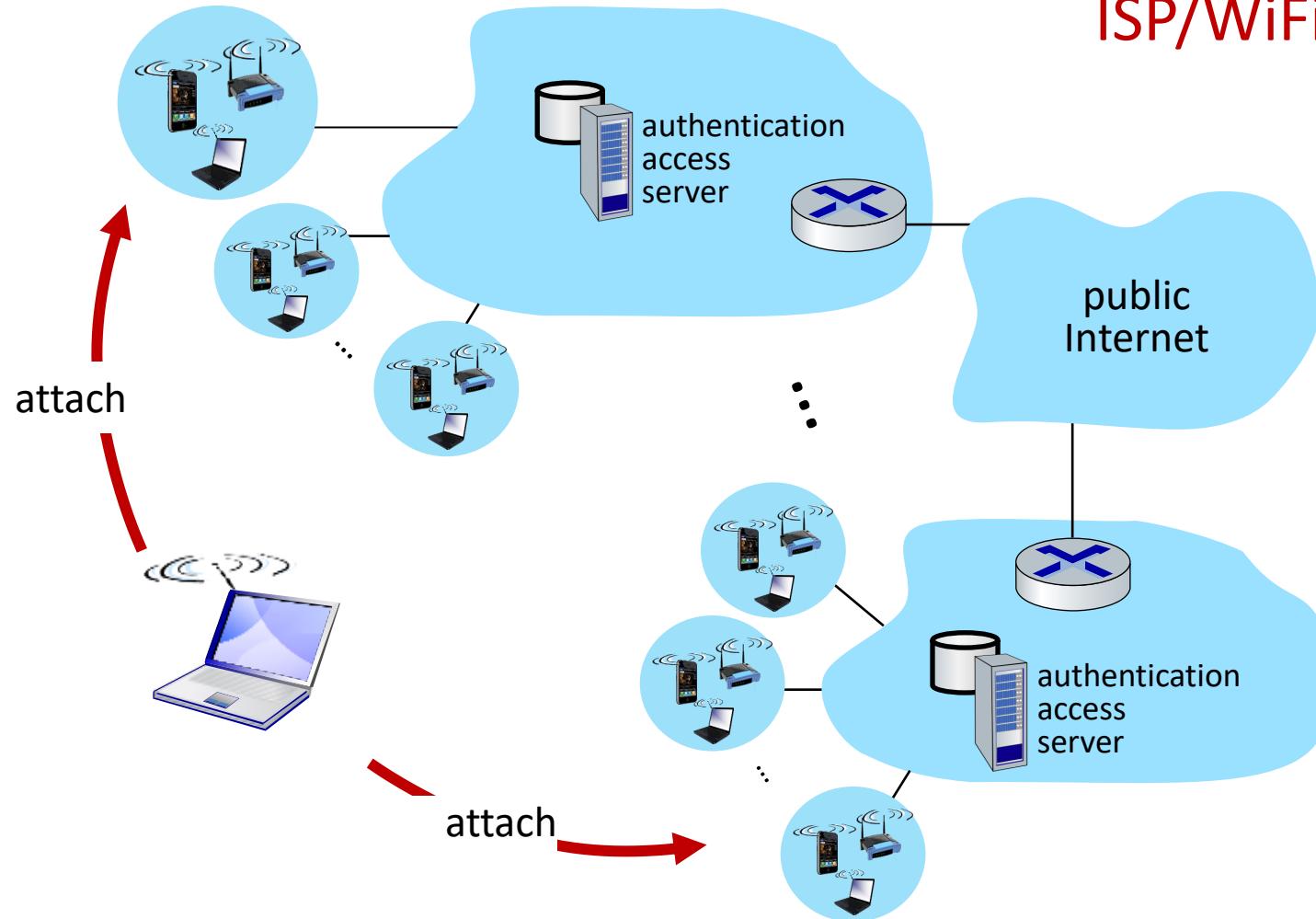
## home network:

- (Paid) service plan with cellular provider, e.g., Verizon, Orange
- Home network HSS stores identify & services info

## visited network:

- Any network other than your home network
- Service agreement with other networks: to provide access to visiting mobile

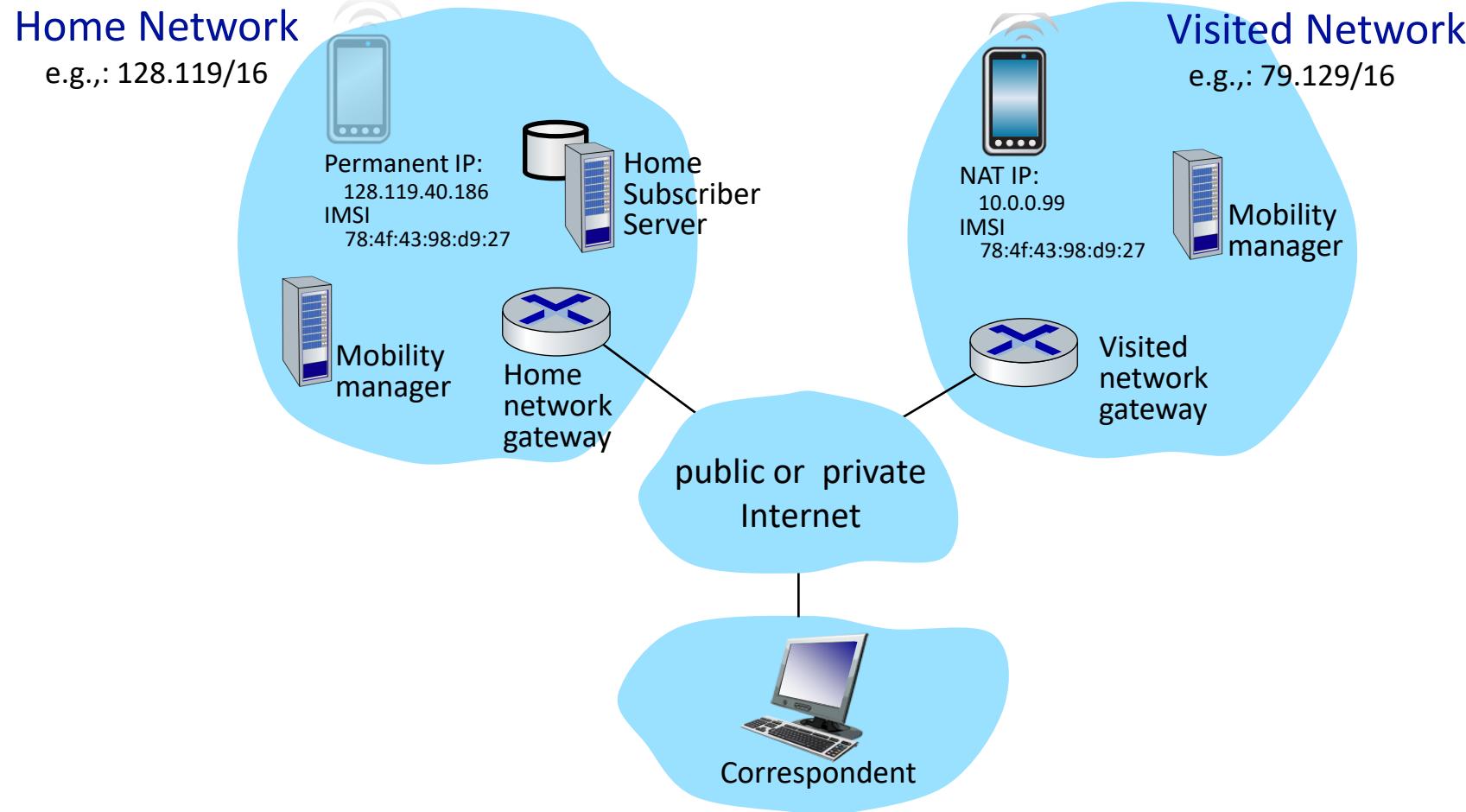
# Home network, visited network: ISP/WiFi



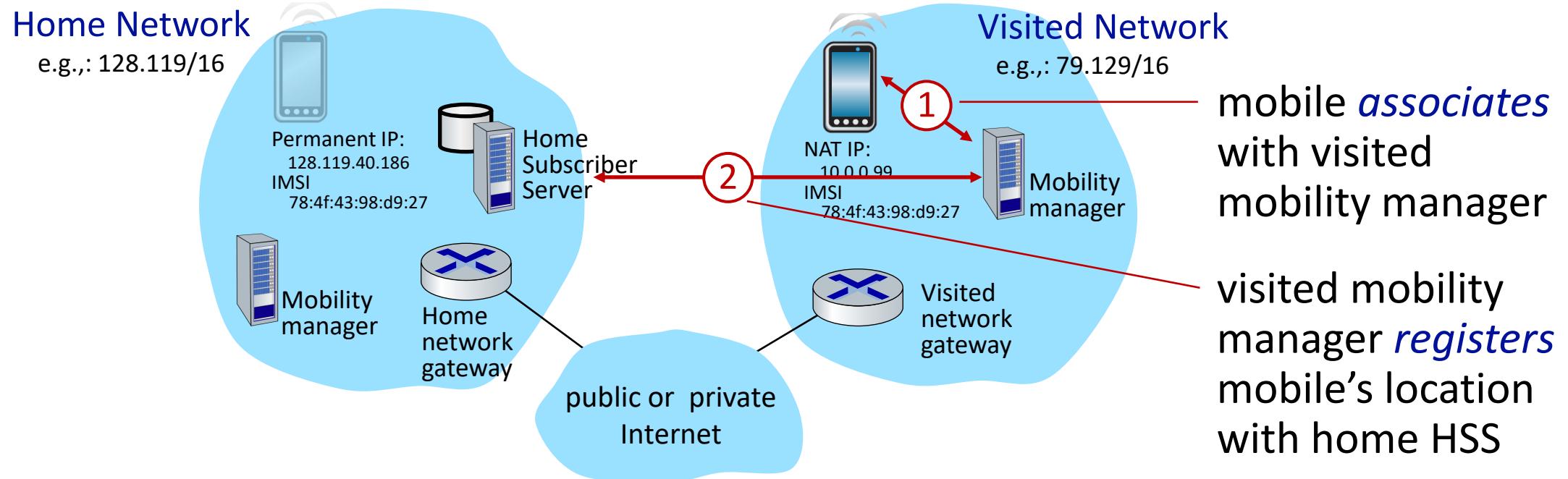
ISP/WiFi: no notion of global “home”

- Credentials from ISP (e.g., username, password) stored on device or with user
- ISPs may have national, international presence
- Different networks: different credentials
  - Some exceptions (e.g., eduroam)
  - Architectures exist (mobile IP) for 4G-like mobility, but not used

# Home network, visited network: generic



# Home network, visited network: generic



End result:

- Visited mobility manager knows about mobile
- Home HSS knows location of mobile

# What is coming next?



- Wireless Networking



- Security



- New Ideas: Evolving Networks



# Course on Computer Communication and Networks

## Lecture 13 Chapter 8; Security and Safety

EDA344/DIT423/LEU062

Lecturer: Hans-Martin Heyn

Based on the book Computer Networking: A Top Down Approach, Jim Kurose, Keith Ross, Addison-Wesley.

# What is coming next?



© Cisco Systems

- Wireless Networking



- Security



- New Ideas: Evolving Networks

# Guest Lecture on Mon. 4th March at 10:00 **VOLVO**

- **Mahboobeh Daftari** is a Security Architect (PKI Lead Architect) at Volvo Trucks, Göteborg.
  - She has worked as Security Expert in the automotive industry for more than 6 years.
- She graduated from Chalmers in 2016 with a M.Sc. in Computer Systems and Networks

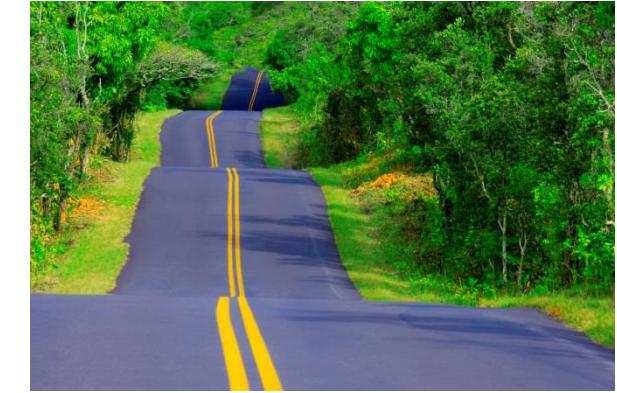


“Communication between vehicles and OEM backend systems is required for many purposes including incident analysis and vehicle software update. A common way to secure end-to-end communication between Electronic Control Units (ECUs) and OEM backend systems is to secure diagnostic service requests and responses between diagnostic tools and the ECUs.

**Unified Diagnostic Services (UDS) protocol is a standard protocol defining two services for enabling security of diagnostics on the application layer:** Security Access (0x27) and Authentication Service (0x29). The overall security of the communication depends on the implementation of the services and the underlying key management which is OEM specific.”

# Roadmap Security and Safety

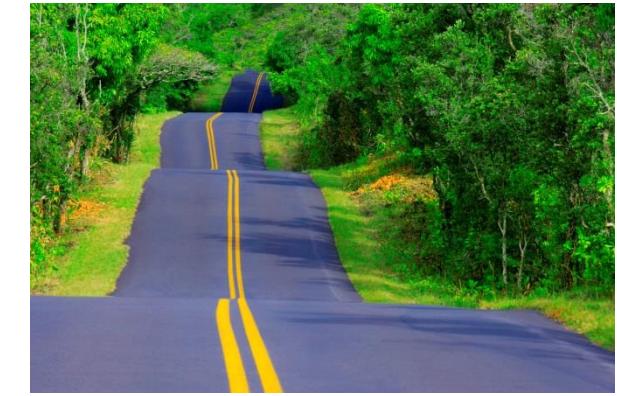
- Principles of network security
  - What is network security?
  - Principles of cryptography
  - Authentication
  - Message Integrity
- Security in practice
  - Securing TCP connections: TLS
  - Network layer security: Ipsec
  - Operational security: Firewalls
- Mission critical communication



# Roadmap Security and Safety



- Principles of network security
  - What is network security?
  - Principles of cryptography
  - Authentication
  - Message Integrity
- Security in practice
  - Securing TCP connections: TLS
  - Network layer security: Ipsec
  - Operational security: Firewalls
- Mission critical communication



# What is security?

---

**Confidentiality**: only sender, intended receiver should “understand” message contents

- sender encrypts message
- receiver decrypts message

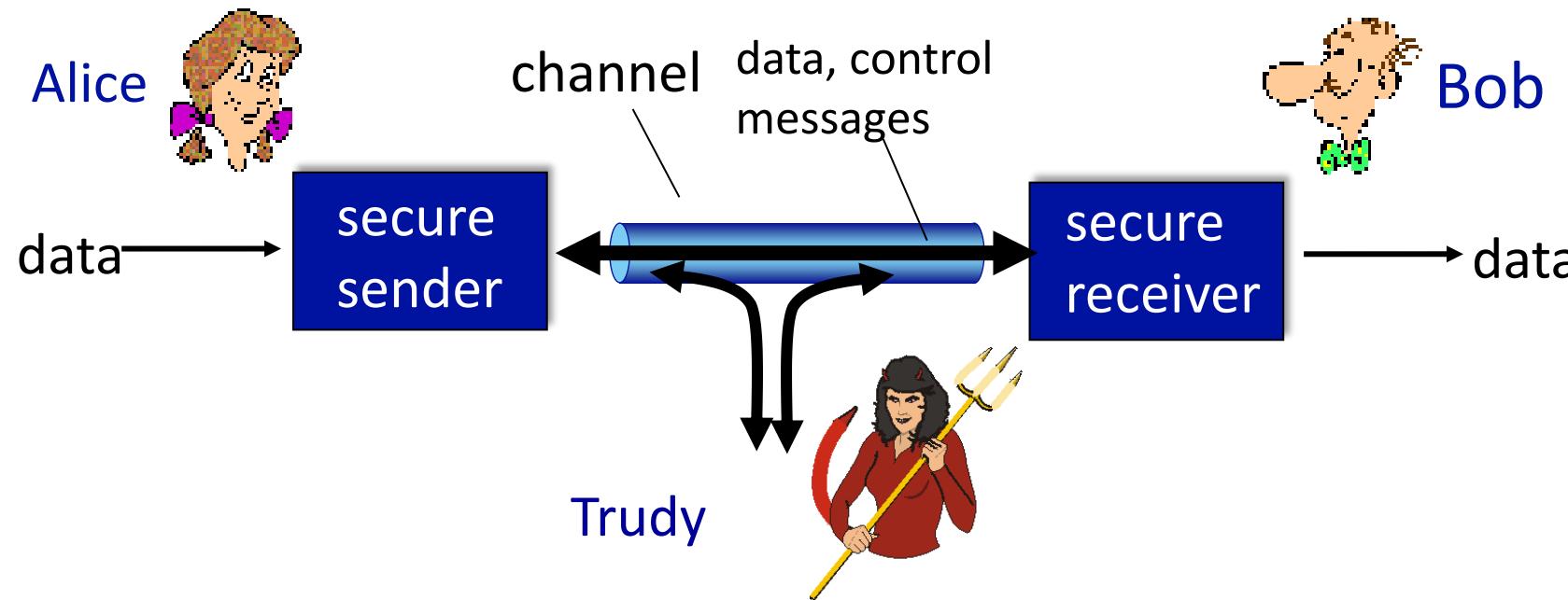
**Integrity**: sender, receiver want to ensure message not altered (in transit, or afterwards) without detection

**Availability**: services must be accessible and available to users

**(Authentication**: sender, receiver want to confirm identity of each other)

# Alice, Bob, Trudy, Eve, and the evil Mallory...

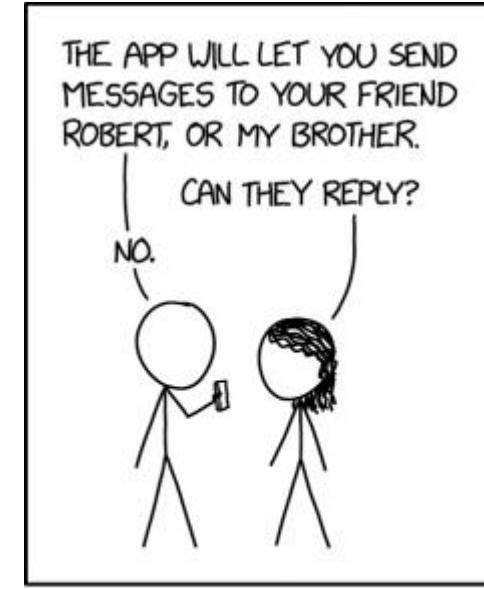
- Well-known in network security world,
- Bob, Alice want to communicate “securely”,
- Trudy (intruder) may intercept, delete, add messages,
- Eve wants to eavesdrop (listens to your messages),
- And Mallory is the really bad guy, a malicious intruder.



# Alice, Bob, Trudy, Eve, and the evil Mallory...

Who might Bob and Alice be?

- ... well, *real-life* Bobs and Alices!
- Web browser/server for electronic transactions (e.g., on-line purchases)
- on-line banking client/server
- DNS servers
- BGP routers exchanging routing table updates
- other examples?



# There are bad people in the Internet!

---

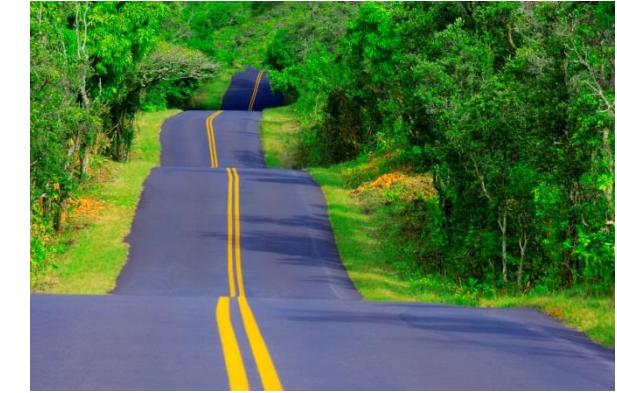
Q: What can a “bad people” do?

A: A lot!

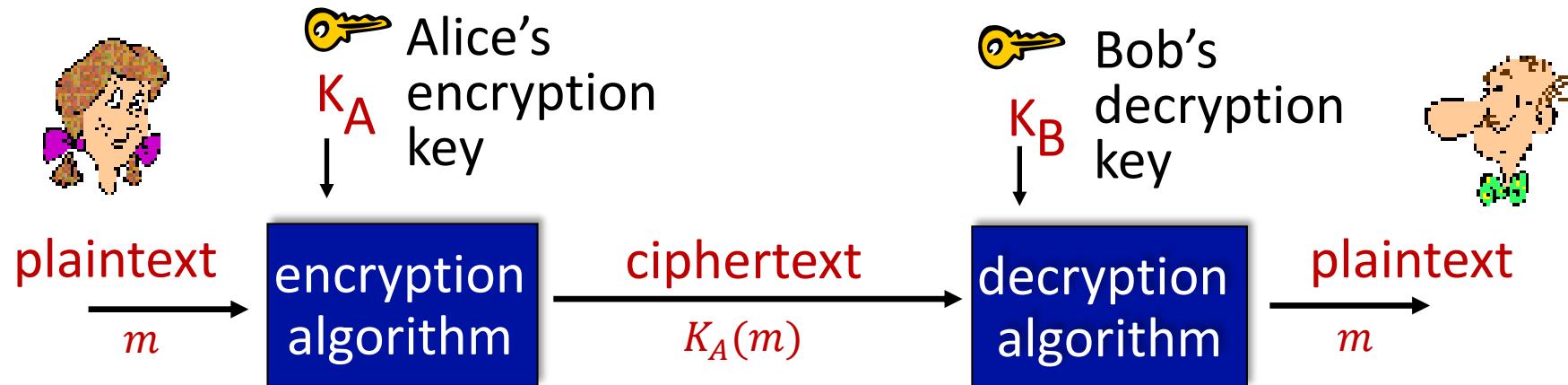
- **Eavesdrop:** intercept messages
- Actively **insert** messages into connection
- **Impersonation:** can fake (spoof) source address in packet (or any field in packet)
- **Hijacking:** “take over” ongoing connection by removing sender or receiver, inserting himself in place
- **Denial of service:** prevent service from being used by others (e.g., by overloading resources)

# Roadmap Security and Safety

- Principles of network security
  - What is network security?
  - Principles of cryptography
  - Authentication
  - Message Integrity
- Security in practice
  - Securing TCP connections: TLS
  - Network layer security: Ipsec
  - Operational security: Firewalls
- Mission critical communication



# The language of cryptography

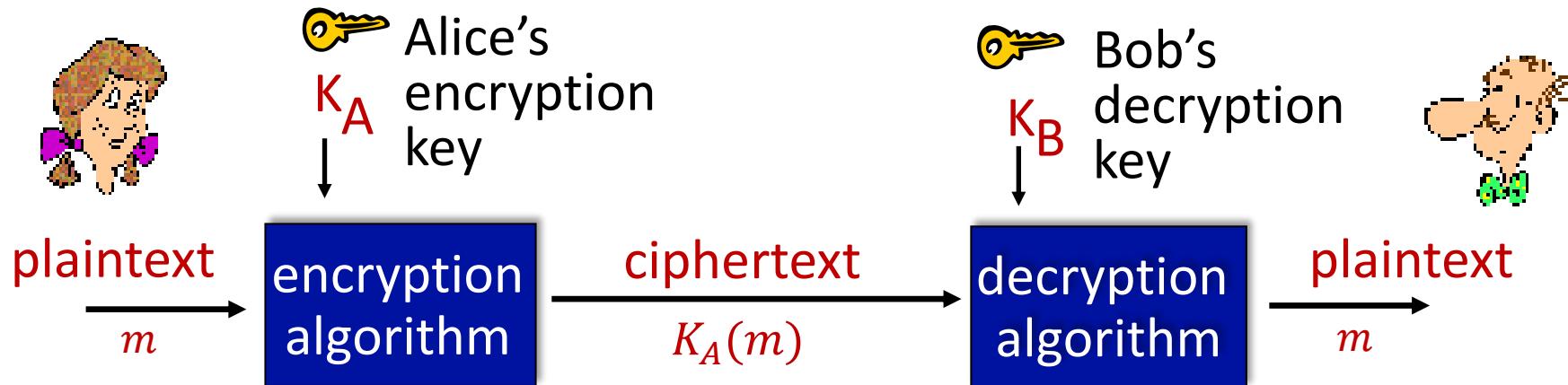


$m$ : plaintext message

$K_A(m)$  : ciphertext, encrypted with key  $K_A$

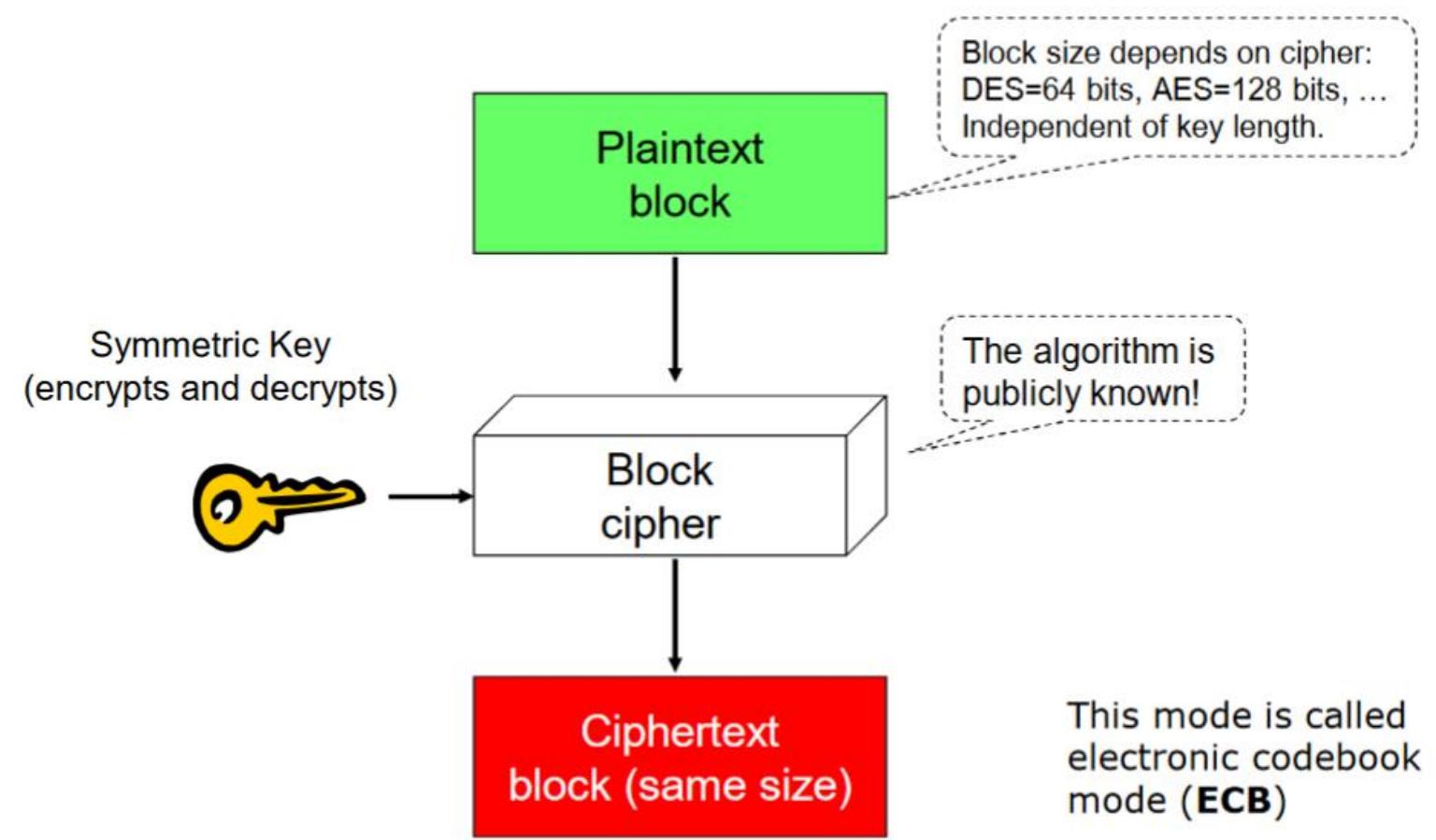
$m = KB(KA(m))$

# Symmetric key cryptography



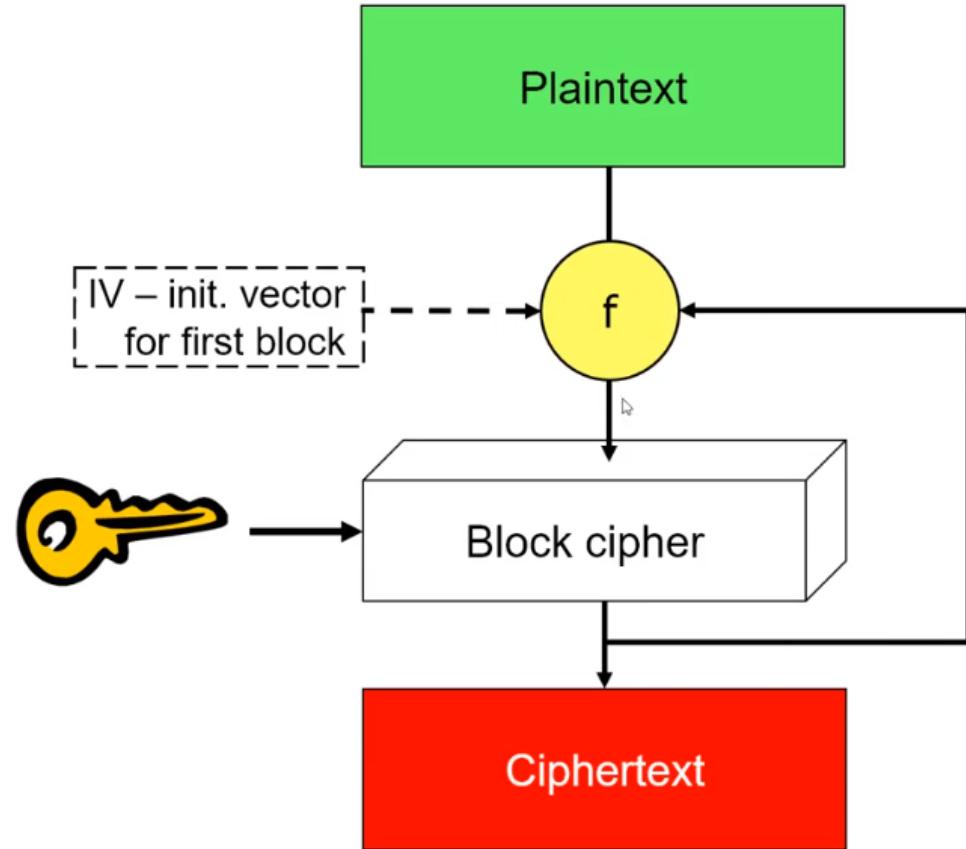
- Symmetric key cryptography: Bob and Alice **share same key**:  $K$ 
  - Key is knowing substitution pattern in an alphabetic substitution cipher
- *Q: How do Bob and Alice agree on key value?*

# ECB - Symmetric key cryptography



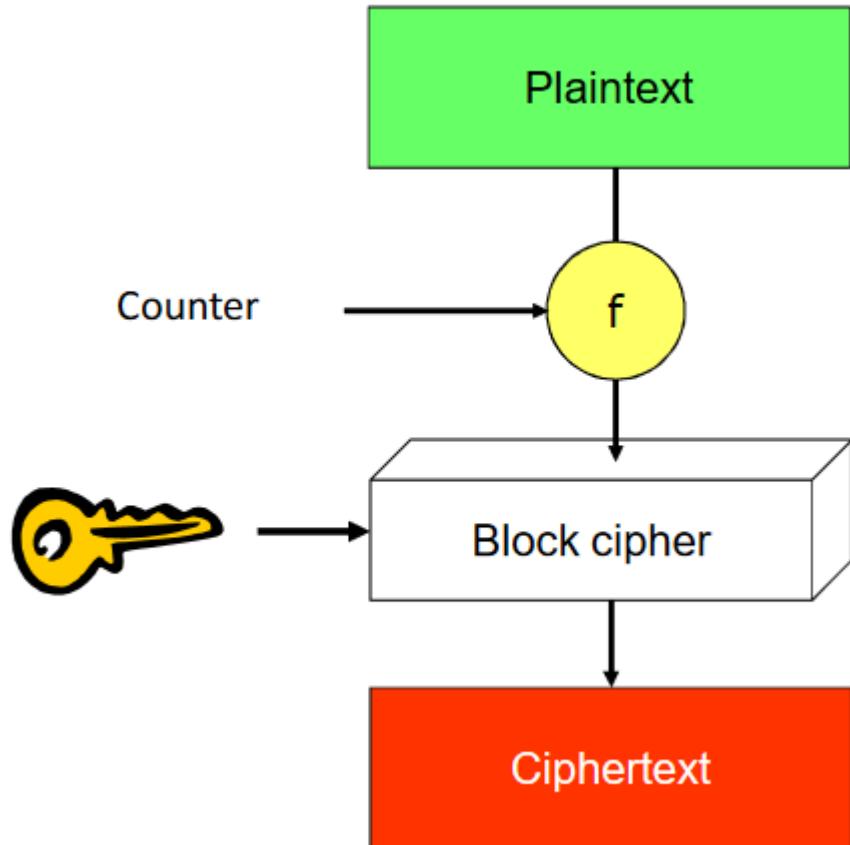
- The algorithm is publicly known (they are even standardised!!)
  - *Don't invent your own cipher-algorithm, it won't work!*
- The key is the secret to security!!
- Problem: What if we have the same plaintext twice?

# CBC - Cipher block chaining mode



- Solution: We can add parts of the previous cipher block to the new cipher
  - Identical blocks now have different ciphertexts.
- May not always be practical.
  - For example in "chains" of information.

# CTR – Counter mode



- We add a (random) counter to the plaintext.
  - Again identical plaintext will now have different ciphertexts, depending on the state of the counter.
- You need to remove the counter for decrypting.

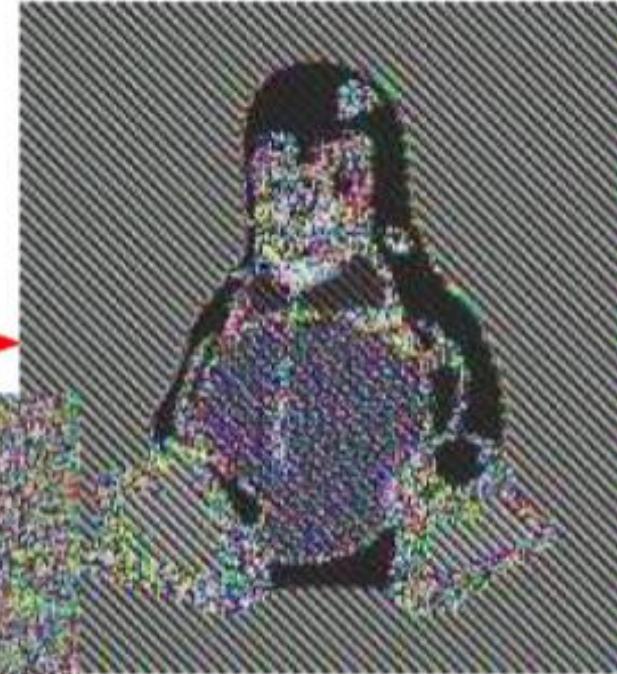
# ECB vs. CBC



CBC



ECB



↑  
Identical blocks  
give identical  
results

© Wikipedia

# Typical symmetric key ciphers

---

- DES (Data Encryption Standard)
  - Designed by IBM 1975, Adopted by NIST\* 1977
  - Old – still widely used
  - Probably more effort is spent on cracking DES than on all other ciphers together
  - Key length is a major problem: 56-bit keys can be cracked in less than a day!
- 3-DES (repeating DES three times with different keys)
  - 3-DES probably secure today but too computationally intensive
- AES (Advanced Encryption Standard)
  - Replaces DES as of 2001, result of an official competition
  - Key lengths: 128, 192 or 256 bits
  - Brute force decryption: if DES takes 1 second, AES-128 takes 149 trillion years,
  - AES-256 would take 1052 years

\*NIST = National Institute of Standards and Technology, US

# Key Length is important!

| Key Length<br>in Bits | Number of Possible Keys                   |
|-----------------------|-------------------------------------------|
| 1                     | 2                                         |
| 2                     | 4                                         |
| 40                    | 1,099,511,627,776                         |
| 56                    | 72,057,594,037,927,900                    |
| 112                   | 5,192,296,858,534,830,000,000,000,000,000 |
| 168                   | 3.74144E+50                               |
| 256                   | 1.15792E+77                               |
| 512                   | 1.3408E+154                               |

WEAK

Strong

# Asymmetric key encryption

## Symmetric key crypto:

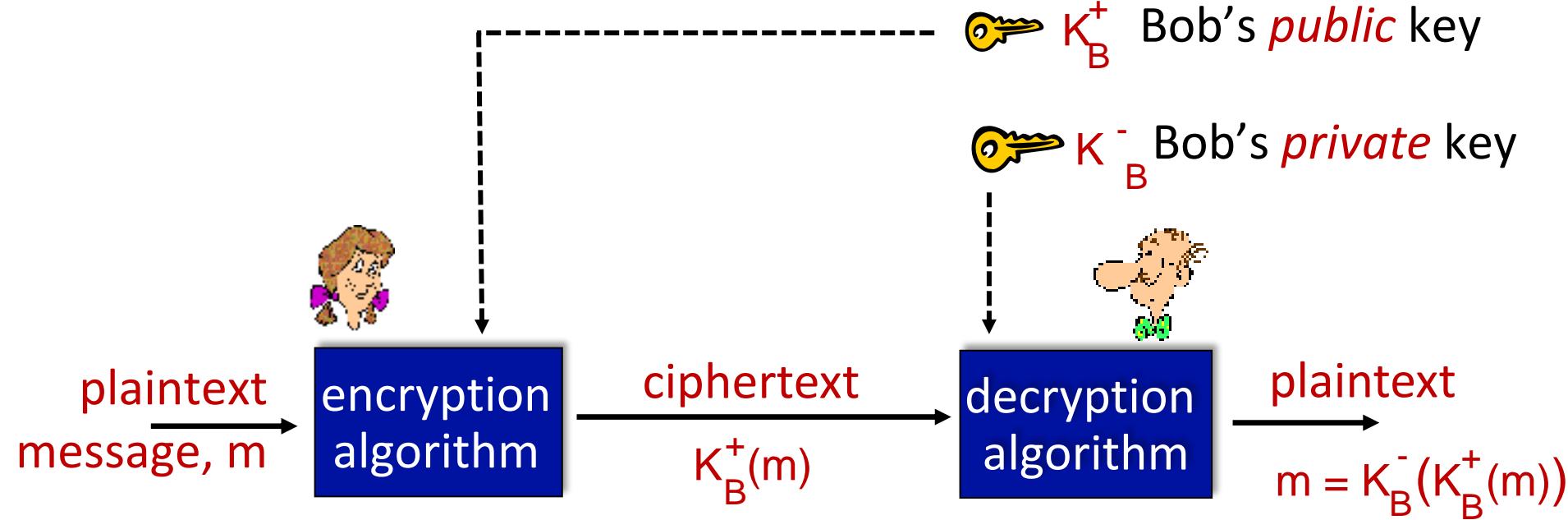
- Requires sender & receiver know shared secret key
- Q: how to agree on key in first place (particularly if never “met”)?

## Public key crypto

- *Radically* different approach [Diffie-Hellman76, RSA78]
- Sender & receiver do *not* share secret key
- *Public* key known to *all*
- *Private* key known only to one party (sender or receiver)



# Asymmetric key encryption



**Wow** - public key cryptography revolutionized 2000-year-old (previously only symmetric key) cryptography!

- similar ideas emerged at roughly same time, independently in US and UK (classified)

# Asymmetric key encryption

One key can be public – the other is kept secret

You decide which key should be public:

1. **Encryption key public:** everyone can send encrypted messages to owner of the private key  
Can be used for **email** and **user authentication**.



2. **Decryption key public:** only one can encrypt, everyone can verify that the secret key has been used.  
Can be used to **sign** documents and data.



# Asymmetric key encryption

---



- One key is used to encrypt, the other to decrypt
- One key can be public – the other is kept secret
- Based on mathematically hard problems
  - RSA – Rivest, Shamir, Adleman (Patented 1983-2000)
    - Factorization of very large numbers
    - Really discovered by GCHQ 1973 but kept secret until 1997...
  - ECC – Elliptic Curve Cryptography (1985)
    - A curve is defined, and keys created from it
    - Standardized curves exist: NIST, IETF, Microsoft, ...
    - Hard to know what curves are secure!
- Slow because of the large numbers involved
  - 1024 bits and up (RSA), 384 bits (ECC)
  - $2^{1024} = 10^{308}$  which means 300 digit numbers
- Use 2048-bit RSA keys in sensitive applications (600 digits)

# RSA widely used algorithm

---

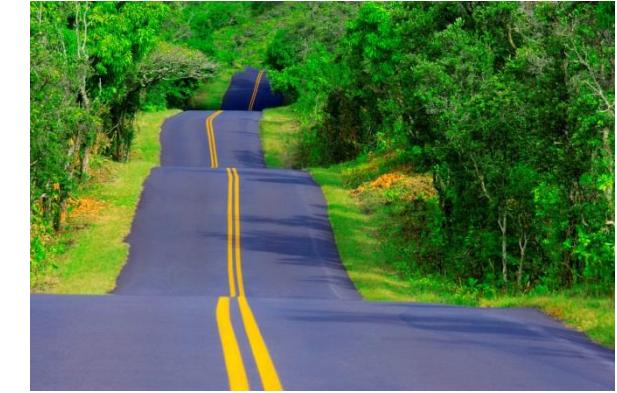
- Exponentiation in RSA is computationally intensive
  - 2048-bit RSA: generation 200 signatures/s (**verification 2000 signatures/s**)
- AES is more than 1000 times faster than RSA
  - Gigabit speed
- Use public key cryptography to agree symmetric crypto keys
- Use symmetric session keys for encrypting data

## Method:

- Bob and Alice use RSA to exchange a symmetric session key  $K_S$
- Example: Bob encrypts symmetric session key with Alice's public key

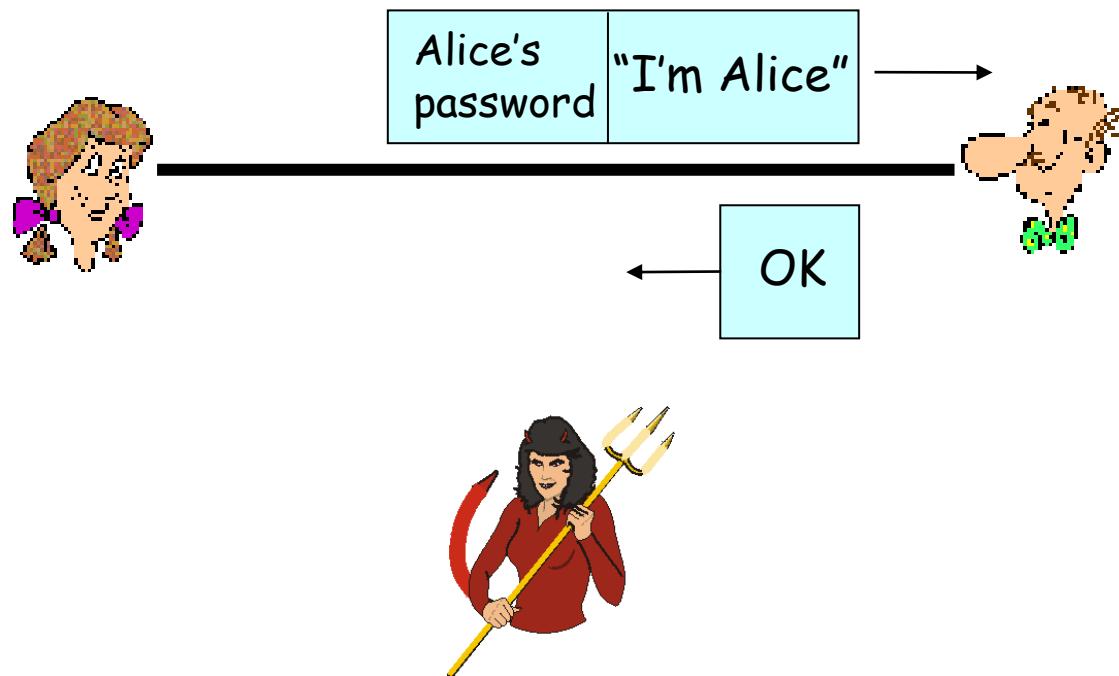
# Roadmap Security and Safety

- Principles of network security
  - What is network security?
  - Principles of cryptography
  - Authentication
  - Message Integrity
- Security in practice
  - Securing TCP connections: TLS
  - Network layer security: Ipsec
  - Operational security: Firewalls
- Mission critical communication



# User Authentication

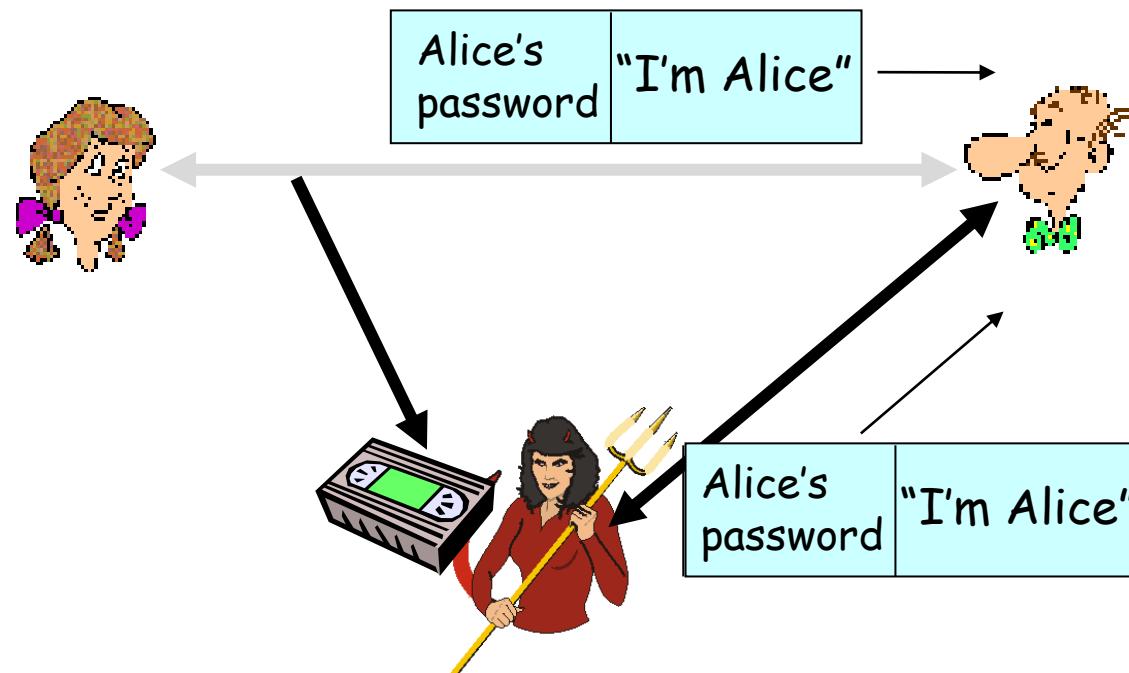
Alice says “I am Alice” and sends her secret password to “prove” it.



What can possibly go wrong here?

# User Authentication

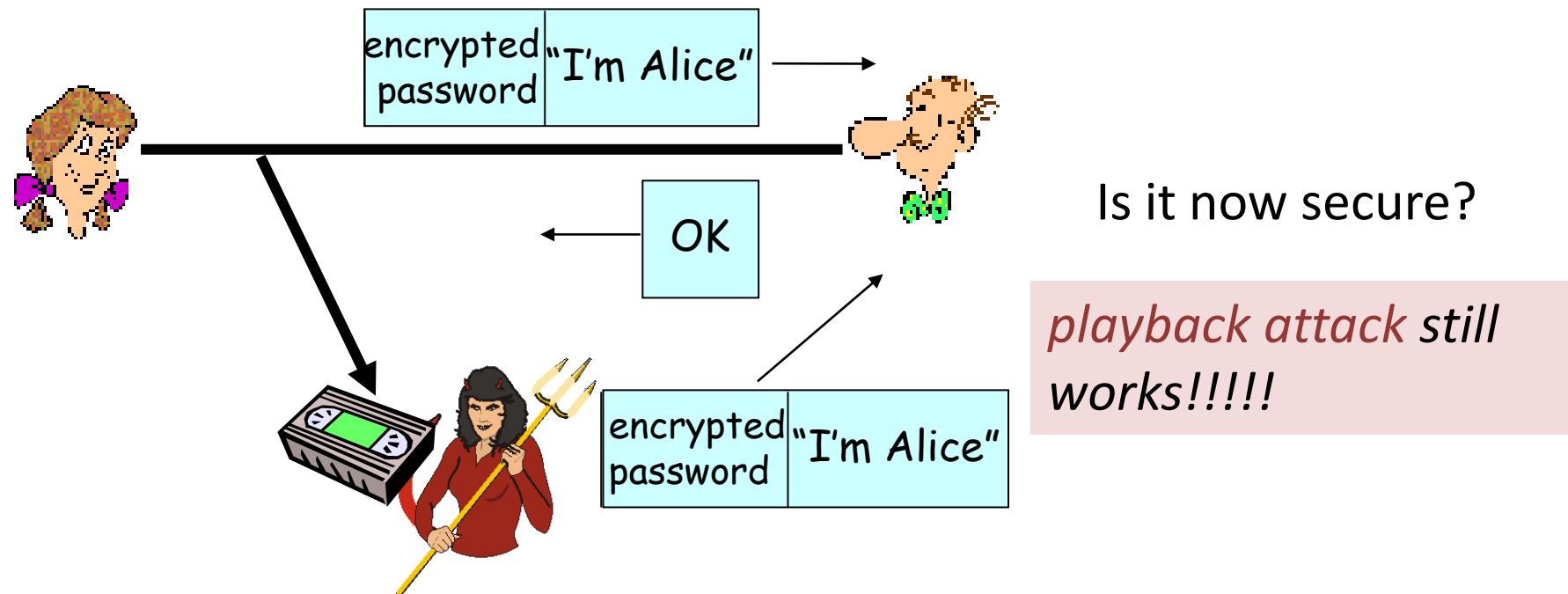
Alice says “I am Alice” and sends her secret password to “prove” it.



*playback attack:* Trudy records Alice’s packet and later plays it back to Bob

# User Authentication

Alice says “I am Alice” and sends her **encrypted** secret password to “prove” it.

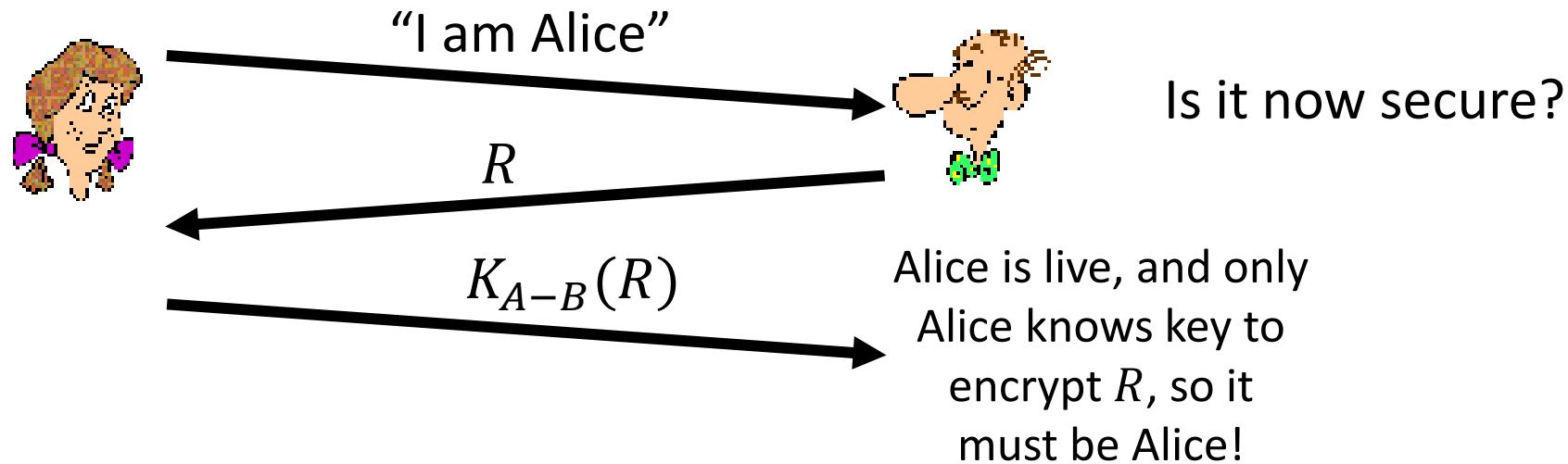


# Challenge response authentication

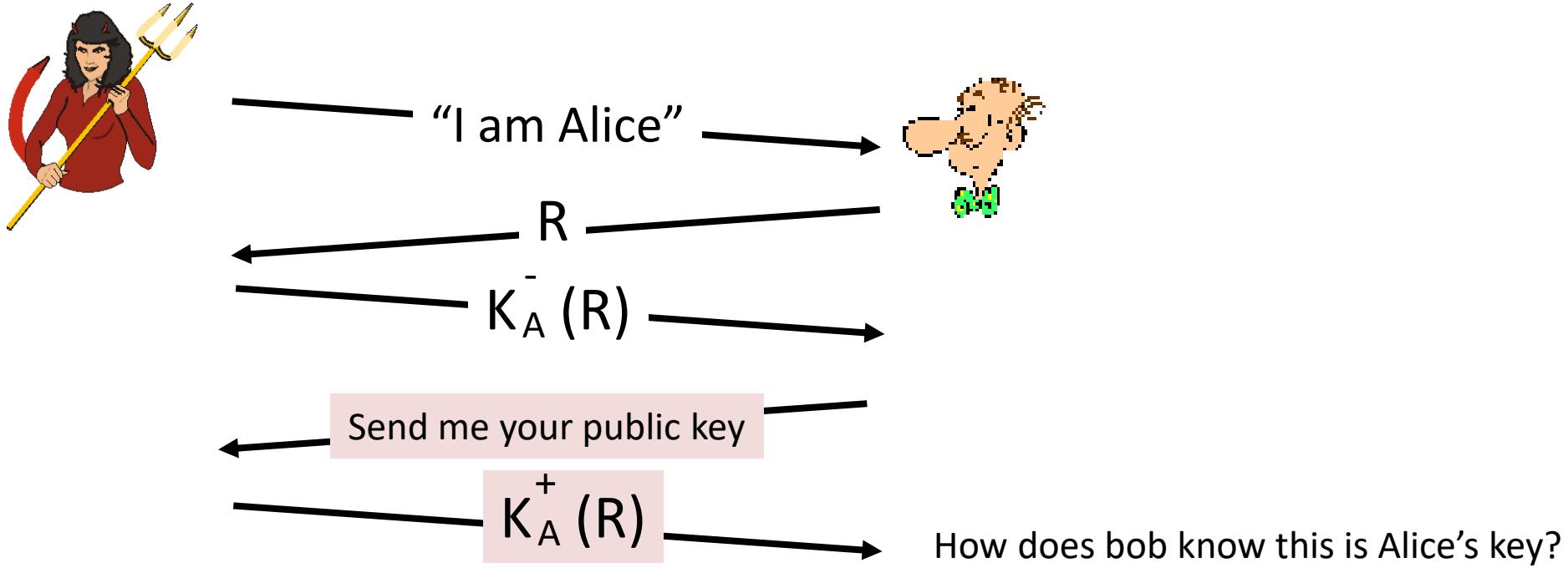
Goal: avoid playback attack

**Solution:** Random number ( $R$ ) used only *once-in-a-lifetime*

1. To prove Alice is “live”, Bob sends Alice **random number**,  $R$ .
2. Alice must return  $R$ , encrypted with shared secret key



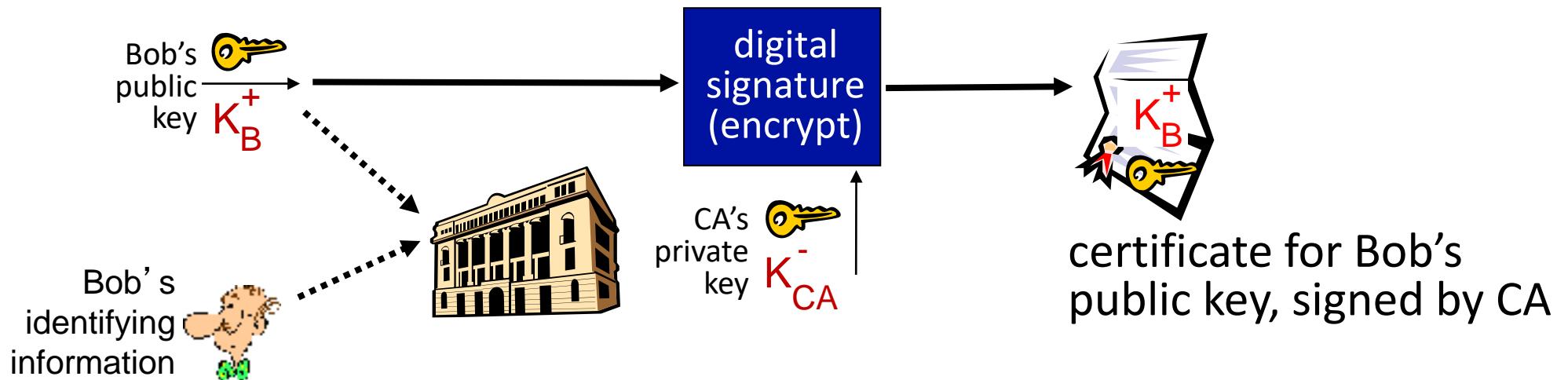
# Authentication with public keys



Public key must be checked

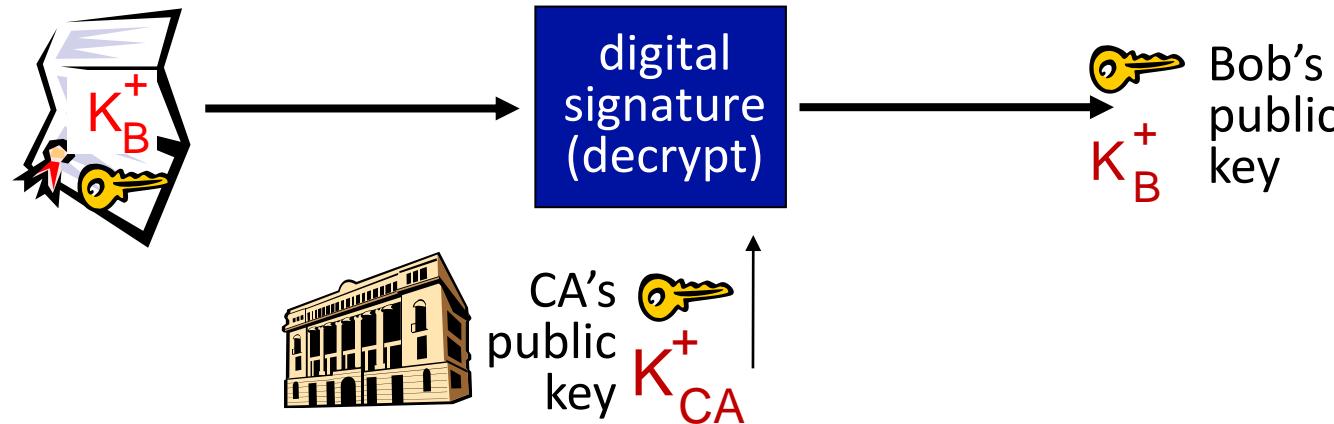
# Public key Certification Authorities (CA)

- **Certification authority (CA):** binds public key to particular entity, E
- Entity (person, website, router) registers its public key
  - CA (certificate authority) creates certificate, binding identity to public key
  - Certificate containing E's public key digitally signed by CA: CA says “this is E's public key”

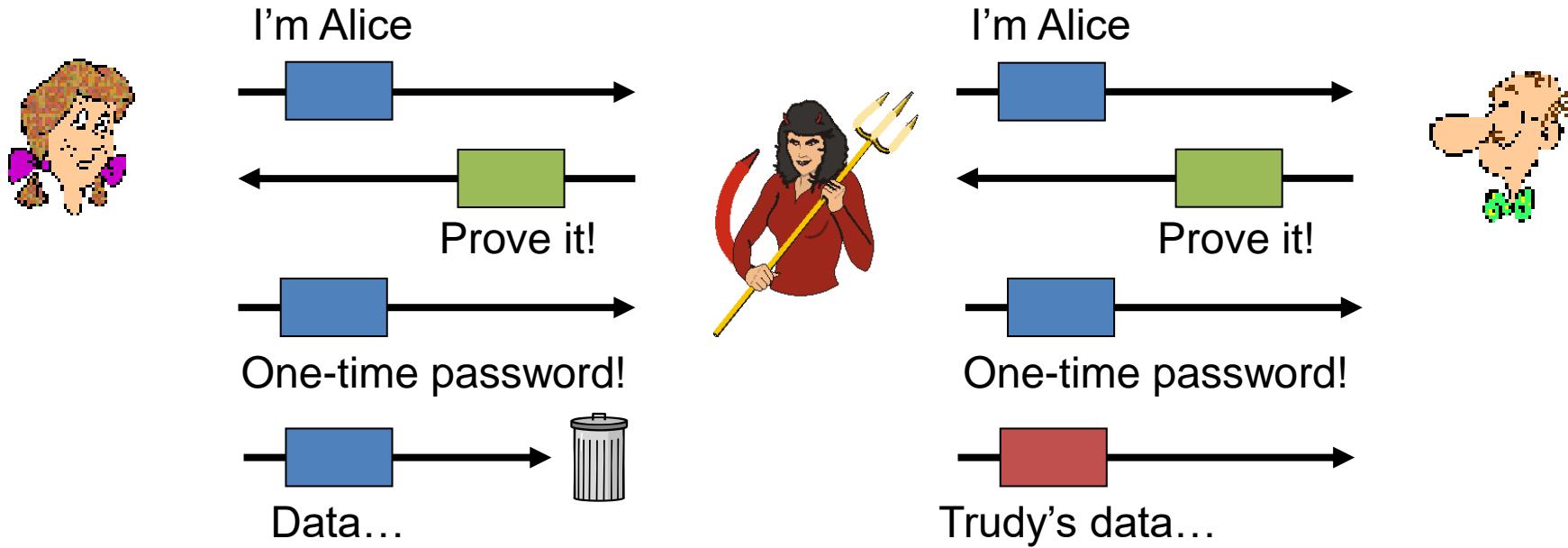


# Public key Certification Authorities (CA)

- When Alice wants Bob's public key:
  - Gets Bob's certificate (Bob or elsewhere)
  - Apply CA's public key to Bob's certificate, get Bob's public key



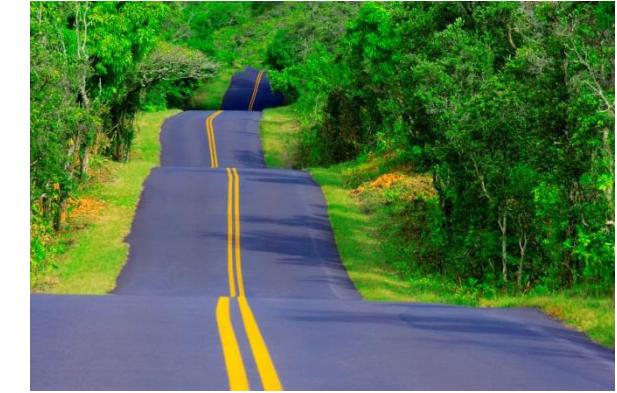
# Man in-the-middle (MITM) attacks



We need packet integrity protection

# Roadmap Security and Safety

- Principles of network security
  - What is network security?
  - Principles of cryptography
  - Authentication
  - Message Integrity
- Security in practice
  - Securing TCP connections: TLS
  - Network layer security: Ipsec
  - Operational security: Firewalls
- Mission critical communication



# Message Integrity

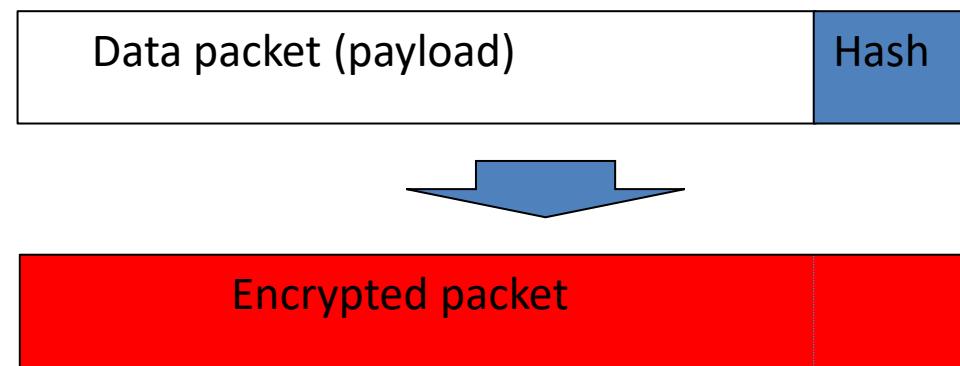
Bob receives message from Alice, wants to ensure:

- Message originally came from Alice
- Message not changed since sent by Alice

Just encryption is not enough to guarantee integrity:

- Contents can be changed even if it is encrypted

Solution: add some kind of checksum (i.e., hash) to the message before it is encrypted:



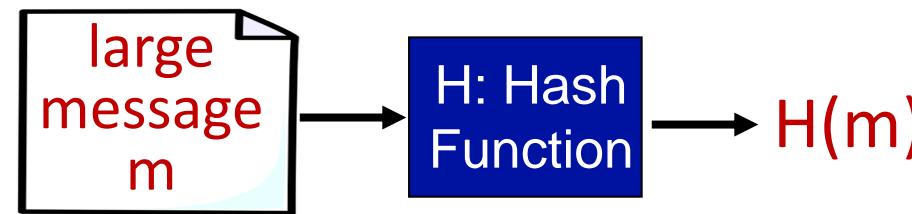
To change the contents, it is necessary to calculate a new hash without knowing the original payload!

# Message digests

---

**Goal:** fixed-length, easy- to-compute digital “fingerprint”

- Apply hash function  $H$  to  $m$ , get fixed size message digest,  $H(m)$

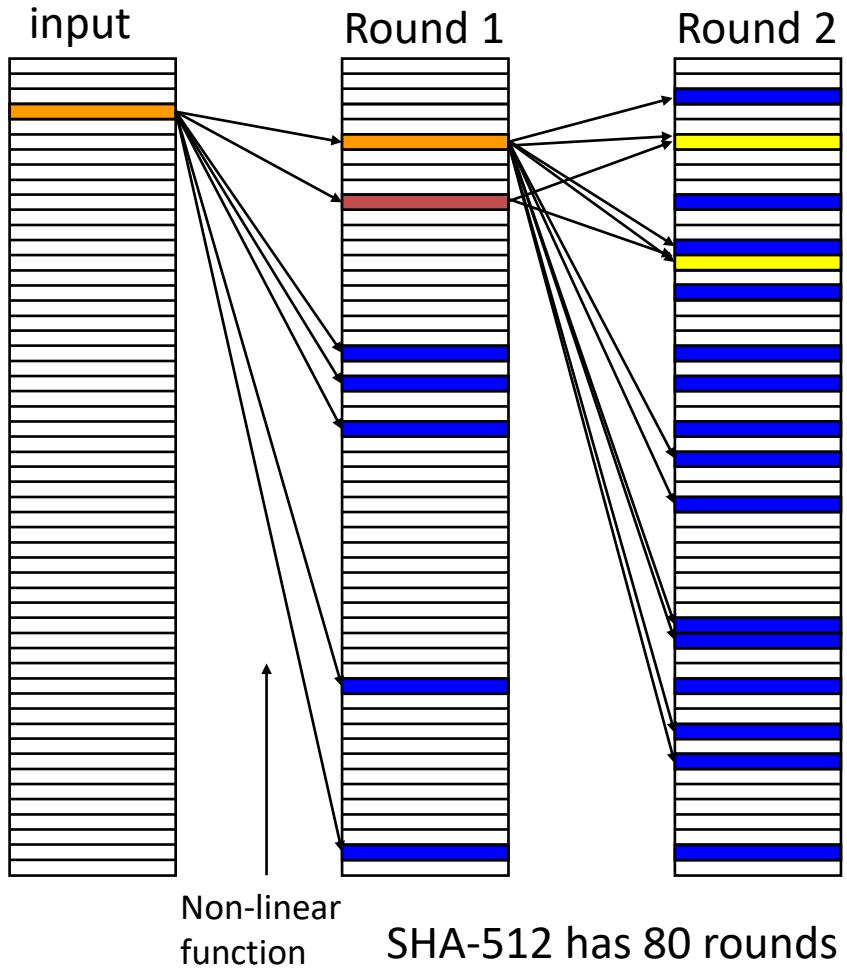


**Hash function properties:**

- Many-to-1
- Produces fixed-size msg digest (fingerprint)
- Given message digest  $x$  it must be computationally infeasible to find  $m$  such that

$$x = H(m)$$

# Hash function algorithms



Even one changed bit gives a **completely different result due to avalanche effect**;

`md5("hello") = 5d41402abc4b2a76b9719d911017c592`

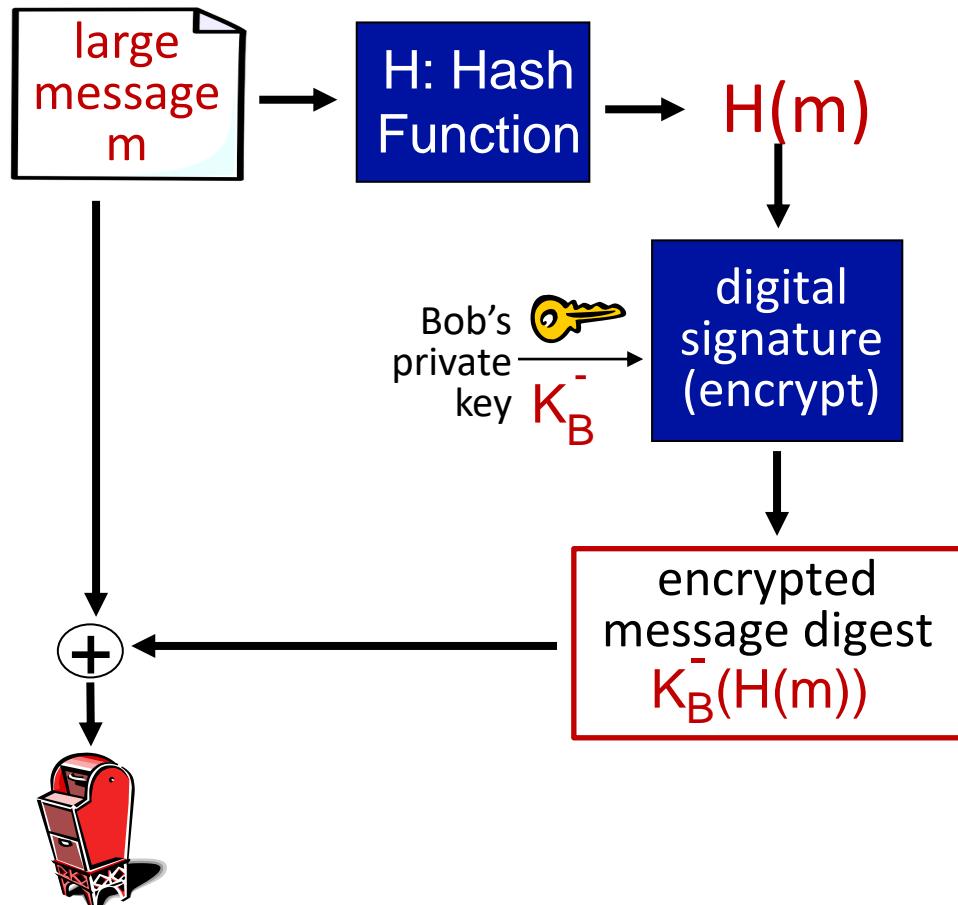
`md5("Hello") = 8b1a9953c4611296a827abf8c47804d7`

- **MD5 hash function widely used (RFC 1321)**
  - computes 128-bit message digest in 4-step process.
  - arbitrary 128-bit string  $x$ , appears difficult to construct msg  $m$  whose MD5 hash is equal to  $x$
- **SHA-1, SHA-256, SHA-512 is also used**
  - US standard [NIST, FIPS PUB 180-1]
  - 20 bytes / 32 bytes / 64 bytes message digest

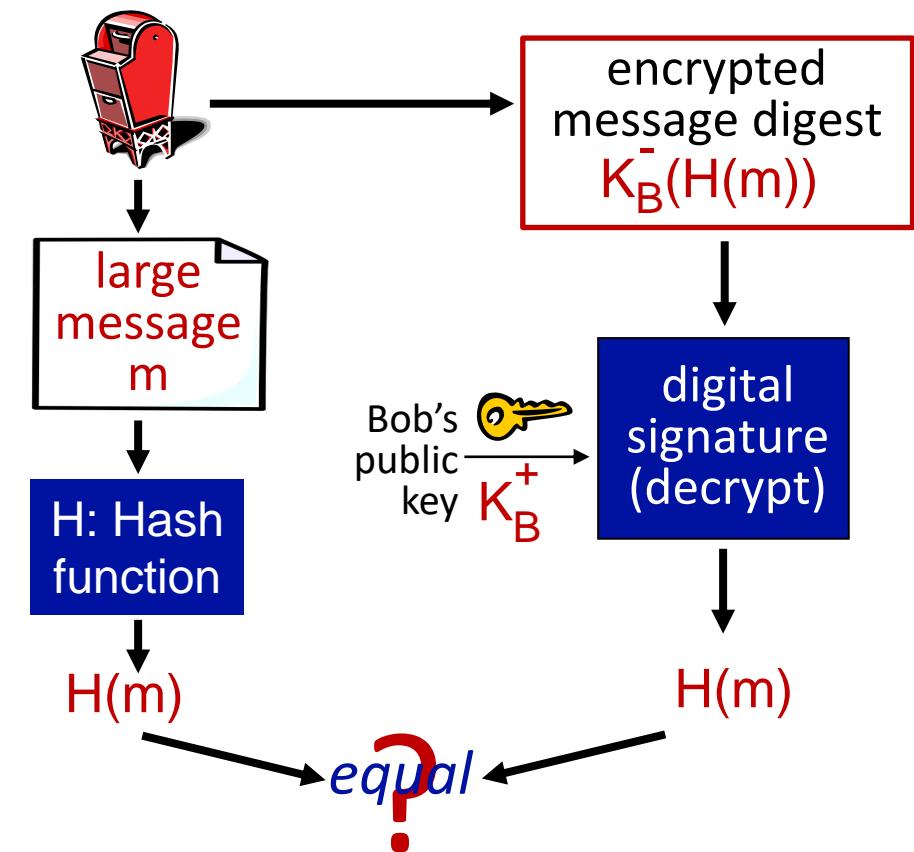
# Digital signature = signed message digest

Example: to identify sender (origin) of emails and guarantee correctness (integrity)

Bob sends digitally signed message:



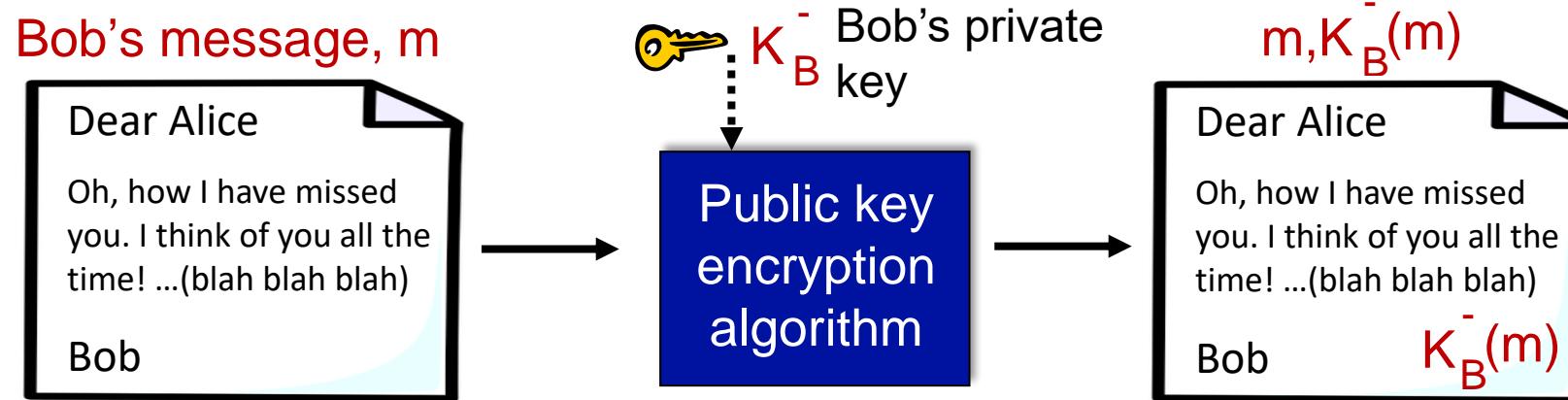
Alice verifies signature, integrity of digitally signed message:



# Digital Signatures

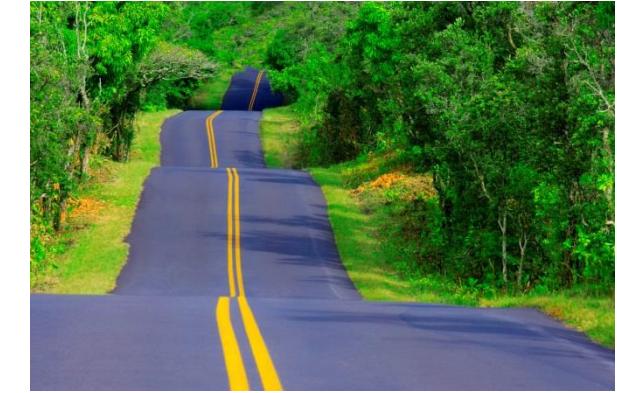
Simple digital signature for message  $m$ :

- Bob signs  $m$  by encrypting with his private key  $K_B$  creating “signed” message,  $K_B(m)$
- Using Bob’s public key,  $K_B(m)$  anyone can verify the signature



# Roadmap Security and Safety

- Principles of network security
  - What is network security?
  - Principles of cryptography
  - Authentication
  - Message Integrity
- Security in practice
  - Securing TCP connections: TLS
  - Network layer security: Ipsec
  - Operational security: Firewalls
- Mission critical communication



# Transport-layer security (TLS)

---

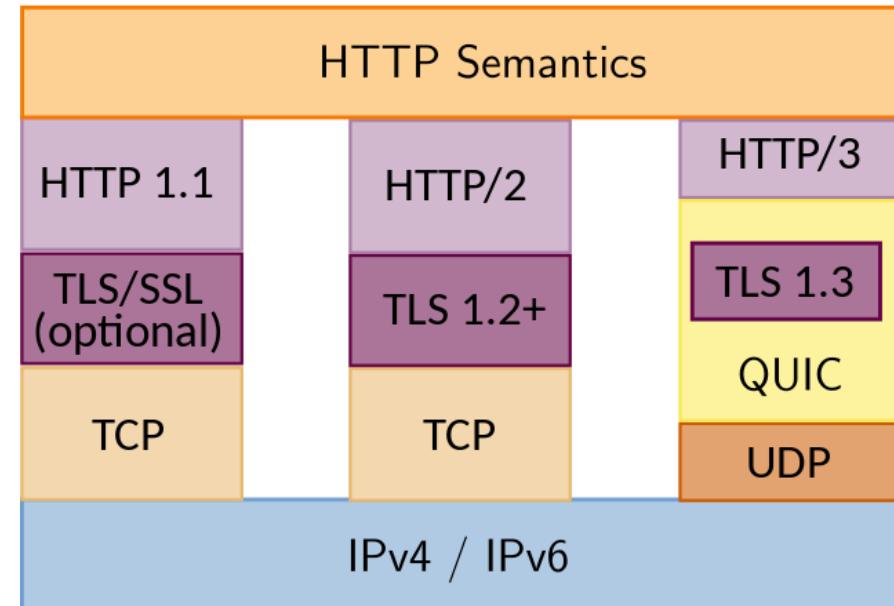
- Widely deployed security protocol above the transport layer
    - supported by almost all browsers, web servers: https (port 443)
  - Provides:
    - Confidentiality: via *symmetric encryption*
    - Integrity: via *cryptographic hashing*
    - Authentication: via *public key cryptography*
  - History:
    - early research, implementation: secure network programming, secure sockets
    - secure socket layer (SSL) deprecated [2015]
    - TLS 1.3: RFC 8846 [2018]
- 
- all techniques we have studied!*

# Transport-layer security (TLS)

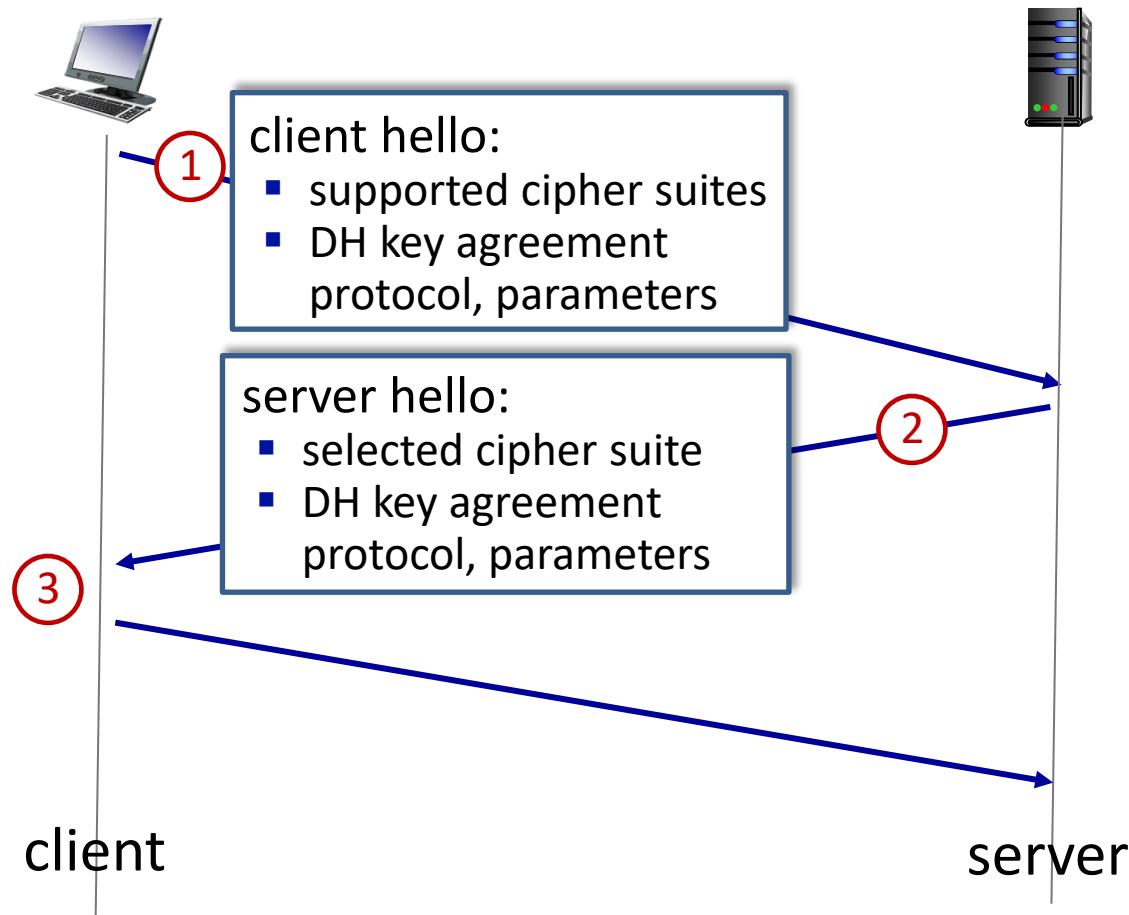
We've seen the “pieces” already:

- **Handshake:** Alice, Bob use their certificates, private keys to authenticate each other, exchange or create shared secret
- **Key derivation:** Alice, Bob use shared secret to derive set of keys
- **Data transfer:** stream data transfer: data as a series of records
  - not just one-time transactions
- **Connection closure:** special messages to securely close connection

TLS provides an API that *any* application can use an HTTP view of TLS:



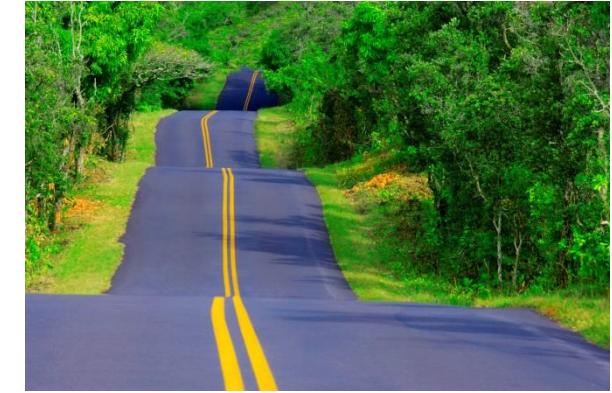
# TLS 1.3: Handshake



- ① client TLS hello msg:
  - guesses key agreement protocol, parameters
  - indicates cipher suites it supports
- ② server TLS hello msg chooses
  - key agreement protocol, parameters
  - cipher suite
  - server-signed certificate
- ③ client:
  - checks server certificate
  - generates key
  - can now make application request (e.g., HTTPS GET)

# Roadmap Security and Safety

- - Principles of network security
    - What is network security?
    - Principles of cryptography
    - Authentication
    - Message Integrity
  - Security in practice
    - Securing TCP connections: TLS
    - Network layer security: Ipsec
    - Operational security: Firewalls
  - Mission critical communication



# IP Sec

- Provides datagram-level encryption, authentication, integrity
  - for both user traffic and control traffic (e.g., BGP, DNS messages)
- Two “modes”:



## Transport mode:

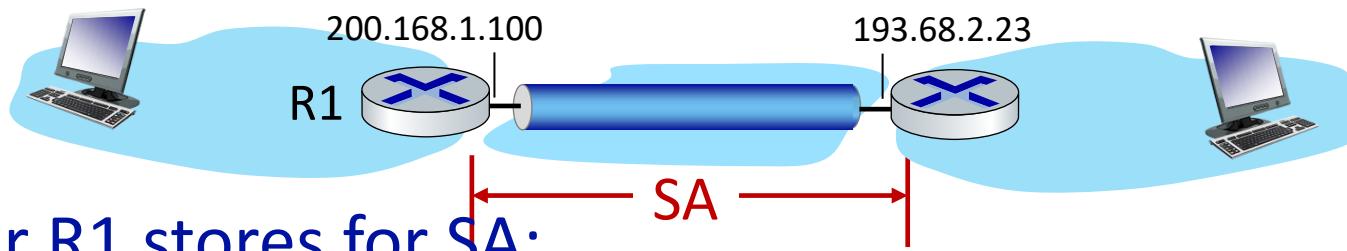
- *only* datagram *payload* is encrypted, authenticated

## Tunnel mode:

- entire datagram is encrypted, authenticated
- encrypted datagram encapsulated in new datagram with new IP header, tunneled to destination

# Security associations (SAs)

- Before sending data, **security association (SA)** established from sending to receiving entity (directional)
- Ending, receiving entities maintain *state information* about SA
  - recall: TCP endpoints also maintain state info
  - IP is connectionless; IPsec is connection-oriented!



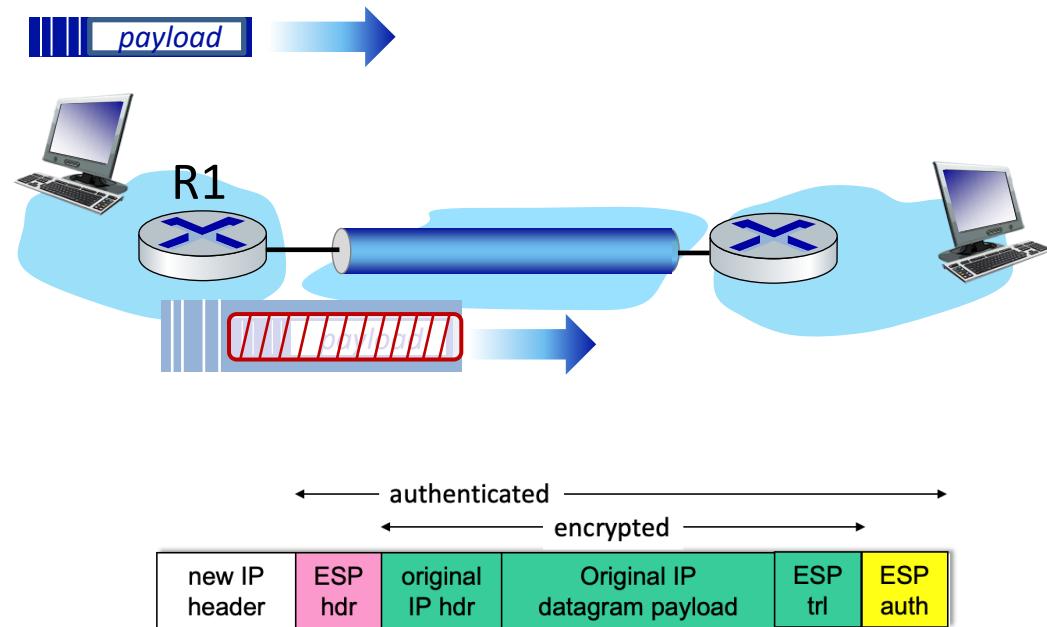
Router R1 stores for SA:

- 32-bit identifier: *Security Parameter Index (SPI)*
- origin SA interface (200.168.1.100)
- destination SA interface (193.68.2.23)
- type of encryption used
- encryption key
- type of integrity check used
- authentication key

# Encapsulation Security Protocol (ESP) tunnel mode

at R1:

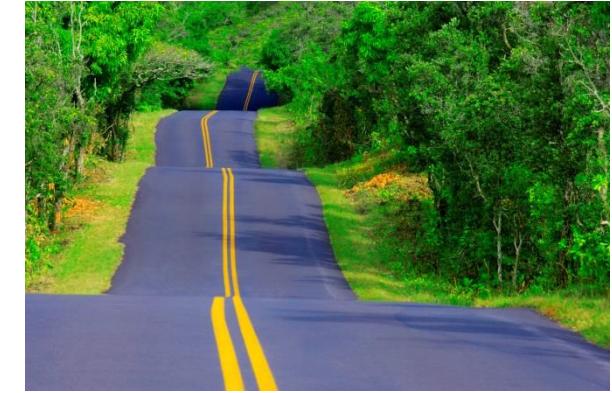
- Appends ESP trailer to original datagram (which includes original header fields!)
- Encrypts result using algorithm & key specified by SA
- Appends ESP header to front of this encrypted quantity
- Creates authentication (message authentication code / MAC) using algorithm and key specified in SA
- Appends MAC forming *payload*
- Creates new IP header, new IP header fields, addresses to tunnel endpoint



- ESP trailer: padding for block ciphers
- ESP header:
  - SPI, so receiving entity knows what to do
  - sequence number, to thwart replay attacks
- MAC in ESP auth field created with shared secret key

# Roadmap Security and Safety

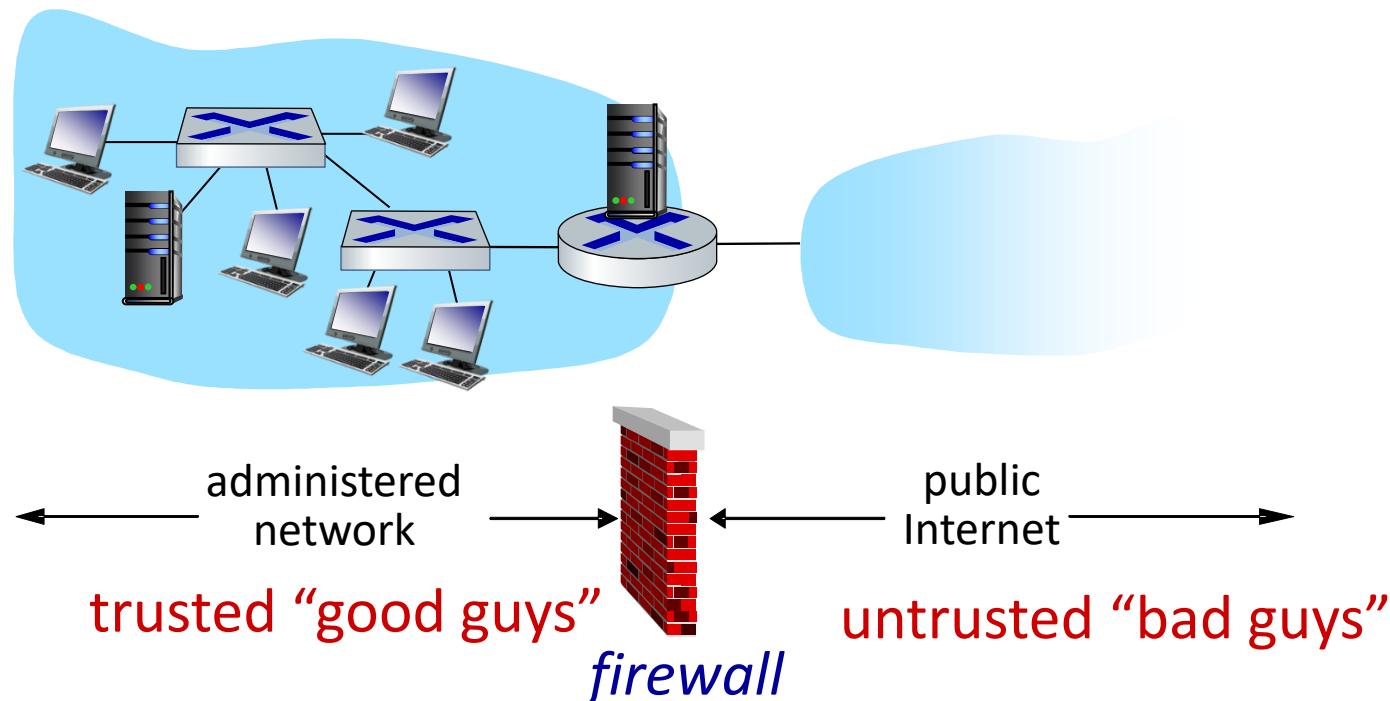
- Principles of network security
  - What is network security?
  - Principles of cryptography
  - Authentication
  - Message Integrity
- Security in practice
  - Securing TCP connections: TLS
  - Network layer security: Ipsec
  - Operational security: Firewalls
- Mission critical communication



# Firewalls

## Firewall

Isolates organization's internal network from larger Internet, allowing some packets to pass, blocking others



# Purpose of Firewalls

---

## Prevent denial of service attacks:

- SYN flooding: attacker establishes many bogus TCP connections, no resources left for “real” connections

## Prevent illegal modification/access of internal data

- E.g., attacker replaces CIA’s homepage with something else

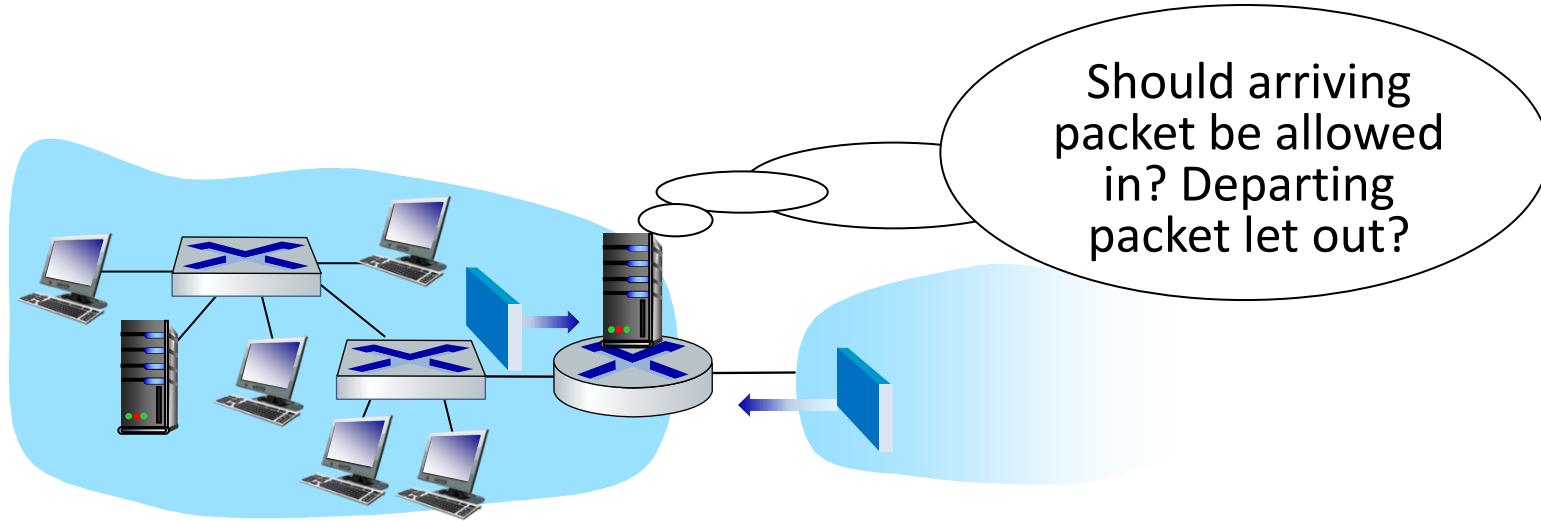
## Allow only authorized access to inside network

- Set of authenticated users/hosts

## Three types of firewalls:

- Stateless packet filters
- Stateful packet filters
- Application gateways

# Stateless packet filtering



- Internal network connected to Internet via router **firewall**
- Filters **packet-by-packet**, decision to forward/drop packet based on:
  - source IP address, destination IP address
  - TCP/UDP source, destination port numbers
  - ICMP message type
  - TCP SYN, ACK bits

# Stateless packet filtering: Examples

**ACL:** table of rules, applied top to bottom to incoming packets:  
(action, condition) pairs

| action | source address          | dest address            | protocol | source port | dest port | flag bit |
|--------|-------------------------|-------------------------|----------|-------------|-----------|----------|
| allow  | 222.22/16               | outside of<br>222.22/16 | TCP      | > 1023      | 80        | any      |
| allow  | outside of<br>222.22/16 | 222.22/16               | TCP      | 80          | > 1023    | ACK      |
| allow  | 222.22/16               | outside of<br>222.22/16 | UDP      | > 1023      | 53        | ---      |
| allow  | outside of<br>222.22/16 | 222.22/16               | UDP      | 53          | > 1023    | ----     |
| deny   | all                     | all                     | all      | all         | all       | all      |

# Stateful packet filtering

---

- *Stateless packet filter*: heavy handed tool

- Admits packets that “make no sense,” e.g., dest port = 80, ACK bit set, even though no TCP connection established:

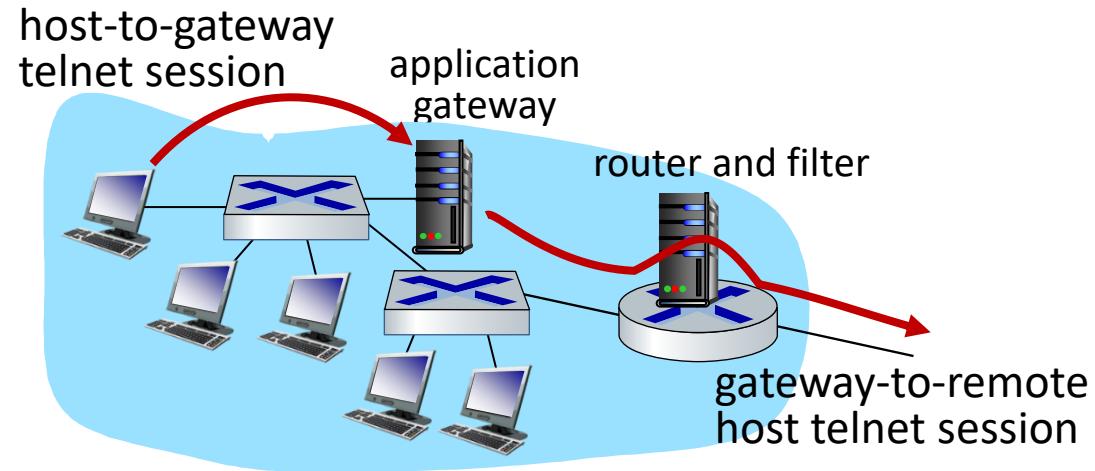
| action | source address          | dest address | protocol | source port | dest port | flag bit |
|--------|-------------------------|--------------|----------|-------------|-----------|----------|
| allow  | outside of<br>222.22/16 | 222.22/16    | TCP      | 80          | > 1023    | ACK      |

- *Stateful packet filter*: track status of every TCP connection

- Track connection setup (SYN), teardown (FIN): determine whether incoming, outgoing packets “makes sense”
- Timeout inactive connections at firewall: no longer admit packets

# Application gateways

- Filter packets on application data as well as on IP/TCP/UDP fields.
- *Example:* allow select internal users to telnet outside



1. Require all telnet users to telnet through gateway.
2. For authorized users, gateway sets up telnet connection to dest host
  - Gateway relays data between 2 connections
3. Router filter blocks all telnet connections not originating from gateway

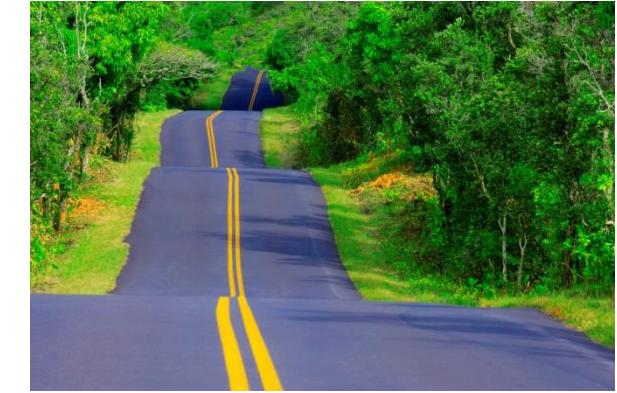
# Limitations of firewalls & gateways

---

- **IP spoofing:** router can't know if data "really" comes from claimed source
- If multiple apps need special treatment, each has own app. gateway
- Client software must know how to contact gateway
  - e.g., must set IP address of proxy in Web browser
- Filters often use all or nothing policy for UDP
- **Tradeoff:** degree of communication with outside world, level of security
- Many highly protected sites still suffer from attacks

# Roadmap Security and Safety

- Principles of network security
  - What is network security?
  - Principles of cryptography
  - Authentication
  - Message Integrity
- Security in practice
  - Securing TCP connections: TLS
  - Network layer security: Ipsec
  - Operational security: Firewalls
- Mission critical communication



# Mission critical communication

## ■ What is mission critical communication?

- Public safety: Communicate warnings, law-enforcement, disaster relief functions (fire brigade), defense.
- Railway, (self-driving) cars: Contact to driver, signaling, position information, accident-prevention, brake control.
- Airplanes: Basically everything...
- Power grid: Sensor information, load balancing, switching, billing.
- Medical: Remote doctors, medical imaging, medical devices (automatic drug administration).
- ....



# Mission critical communication

- Requirements towards mission critical communication depends on use case.
- Requirements base on detailed safety / hazard analysis, but generally:
  - Stringent performance requirements,
  - Availability guarantees,
  - Guarantees on worst-case transmission time,
  - And security / integrity of communication.



What could make a system unavailable?

- Natural disaster
- Software Bug or Hardware Failure
- Lack of maintenance
- Cyber-attack
- Under-dimension / Overload
- Alien Invasion?

# Strategies for mission critical communication

---

- Use of well proven technology
  - For example, automotive industry widely uses CAN bus (Controller Area Network)
    - CAN 1.0: 1987, ca. 40kbit/s, CAN 2.0: 1991, ca. 125 kbit/s, CAN FD: 2012, up to 5 Mbit/s
- Reliability more important than high speed
  - For example, use of deterministic channel sharing (we know always when which device will be allowed to send) instead of random based channel access protocols.
    - We lose bandwidth, but we gain a guarantee when a transmission will be finished in the worst case.
- Communication channel redundancy
  - Having several independent(!!!) communication channels establishes redundancy.
    - Always remember that redundancy in communication only provides additional reliability if (and only if) the individual communication channels are independent from each other.
    - I.e., a failure in one communication channel must not cause the failure of the redundant channel.

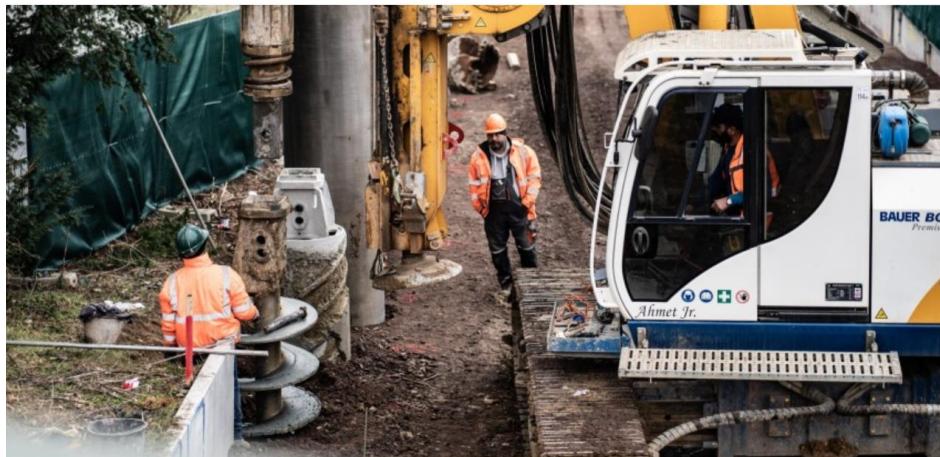
# How to not do redundant communication

INTERNETSTÖRUNG

Bahn hat Glasfaserkabel der Telekom angebohrt

VON JONAS JANSEN, DÜSSELDORF - AKTUALISIERT AM 15.02.2023 - 12:53

faz.net



Flight cancellations due to a cut fiber optic cable: The fault lies with Lufthansa

A digger single-handedly paralyzes Lufthansa. This is not the fault of the excavator operator, but an embarrassing resilience meltdown, comments Susanne Nolte.

heise.de

AIRPLANES AND PASSENGERS STRANDED

"Disruption has far-reaching consequences": Lufthansa's IT systems paralyze Frankfurt Airport

by Angelica Melcher  
February 15, 2023, updated Feb 15, 2023 at 6:27 p.m.



wiwo.de



Lufthansas it-problem påverkade många flygavgångar under onsdagen. Arkivbild. Foto: Ronald Wittek/EPA/TT

svt.se

It-problem hos Lufthansa – kaos på flygplatser

UPPDATERAD 15 FEBRUARI 2023 PUBLICERAD 15 FEBRUARI 2023

Störningar i Lufthansas it-system har orsakat omfattande problem på flygplatser världen över på onsdagen. 200 flygningar till och från flygnavet Frankfurt har ställts in och mer än 100 avgångar har försonats. Orsaken sägs vara en skadad datakabel i Frankfurt, och även

# Summary

---

## Basic techniques.....

- Cryptography (symmetric and public key)
- Message integrity
- End-point authentication

.... used in many different security scenarios

- secure transport (TLS)
- IP sec

Operational security: firewalls and IDS

Mission critical communication



# Course on Computer Communication and Networks

## Lecture 14 Synthesis, Summary, and best of...

EDA344/DIT423/LEU062

Based on the book Computer Networking: A Top Down Approach, Jim Kurose, Keith Ross, Addison-Wesley.

# Important for the exam

---

**When/where:** Wednesday March 13, 14.00-18.00, Johanneberg Campus (without guarantees, check examination dates on chalmers.se and gu.se)

**You may have with you:**

- English-X dictionary
- no calculators, tablets, computers, smartphones, etc. (if/where numbers matter, do rounding)

**Grading**

- 30-40, 41-50, 51-60 (out of 60)= 3, 4, 5 (CTH)
- 30-45, 46-60 (out of 60) = G, VG (GU)

**How to think during summary-study:**

- Work with the training material
- Take a step back, compare/contrast concepts, approaches; explain; ask yourselves: why is this so? / how does it work (or why it does not work)?

**How to think when answering:**

- explain your answers
  - *Rule of thumb:* a fellow student, with the knowledge of our intro lectures, can follow

# You shall pass! The path to pass!!



# Course organisation – Path to Pass



- Before each part of the lecture, a canvas page called "Path to Pass: Reading Recommendation" will appear.

## Path to Pass: Reading Recommendations Part 1

To-do date: 21 Jan at 23:59

This is the reading necessary to prepare yourself on your way towards the exam:

You can read the material before or after the lecture, whatever you prefer.

| ("Part #", as in the lectures modules) | 8th Edition,<br>careful study | 8th Edition,<br>quick reading | 7th Edition,<br>careful study | 7th Edition,<br>quick reading |
|----------------------------------------|-------------------------------|-------------------------------|-------------------------------|-------------------------------|
| Part1: Intro                           | Sections 1.2 - 1.5            | The rest of Chapter 1         | the same                      | the same                      |

*Careful study: It means you must study and understand the material so that you can explain it in your own words.*

*Quick reading: It means you can read quickly over it. You can study it in more detail if you are interested.*

- You can read the provided Sections in the book before or after the lecture, whatever you prefer.
- The to-do-date is a (very very very) strong recommendation to have finished reading the Sections by that date.
  - Study continuously! **Under no circumstances try to panic-read everything the week before the exam! It will not work!**

# Course organisation – Path to Pass



- Similar to the lectures, there will be a path to pass: exercises page appearing about one week before the exercise session.

## Path to Pass: Exercises Part 1 & 2

To-do date: 28 Jan at 23:59

You are expected to have tried to solve the exercises before the exercise sessions. If you cannot find the solution, you can get guidance during the exercise sessions.

We will not publish step-by-step solutions to the book exercises. If you need help, you can ask during the exercise sessions which are offered twice a week.

You are also strongly recommended to try the [interactive exercises page at the companion site of the book](#).

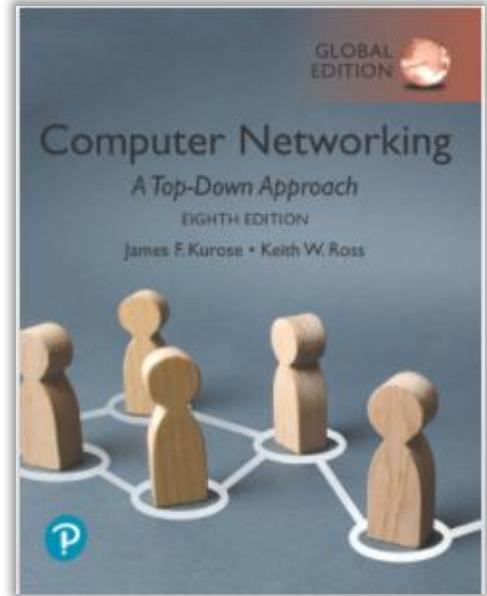
| ("Part #", as in the lectures modules) | 8th Edition                                                                                          | 7th Edition                                                                                                                             |
|----------------------------------------|------------------------------------------------------------------------------------------------------|-----------------------------------------------------------------------------------------------------------------------------------------|
| Part 1: Introduction                   | Chapter 1 Exercises 6, 13, 22, 23, 25<br>Additional nice-to-solve: Chapter 1, Exercises 2, 8, 16, 24 | the same<br>Additional nice-to-solve: the same                                                                                          |
| Part 2: Application Layer              | Chapter 2 Exercises 7, 8, 17, and 21<br>Additional nice-to-solve: Chapter 2, Exercises 1, 3, 19, 20  | Chapter 2 Exercises 7, 8, 15, and 21 (Note: 7 and 8 are slightly different but on the same topic)<br>Additional nice-to-solve: the same |

Additionally, you can try to solve the "nice-to-solve" exercises; but they are not being discussed in the exercise sessions.

- Try to solve the exercises before the sessions. If you get stucked, don't panic! Post your questions in the Discussion board on Canvas. The teachers will take it up then during the exercise session. You can also asked directly in the session.
- Do the exercises continuously! **Under no circumstances start doing all exercises the week before the exam! It will not work! The solutions will not be sufficient for self-study! You MUST come to the exercise sessions to understand them.**

# Recall training material

- ⋮ ▾ Kursinformation
  - ⋮ Computer Communication (Study Period 3)
  - ⋮ ReadingAndExerciseLists
  - ⋮ Useful links
  - ⋮ Exam Information
- ⋮ ▾ Exercises
  - ⋮ ▶ Quizzes (self-assessment)
  - ⋮ ▶ Demos

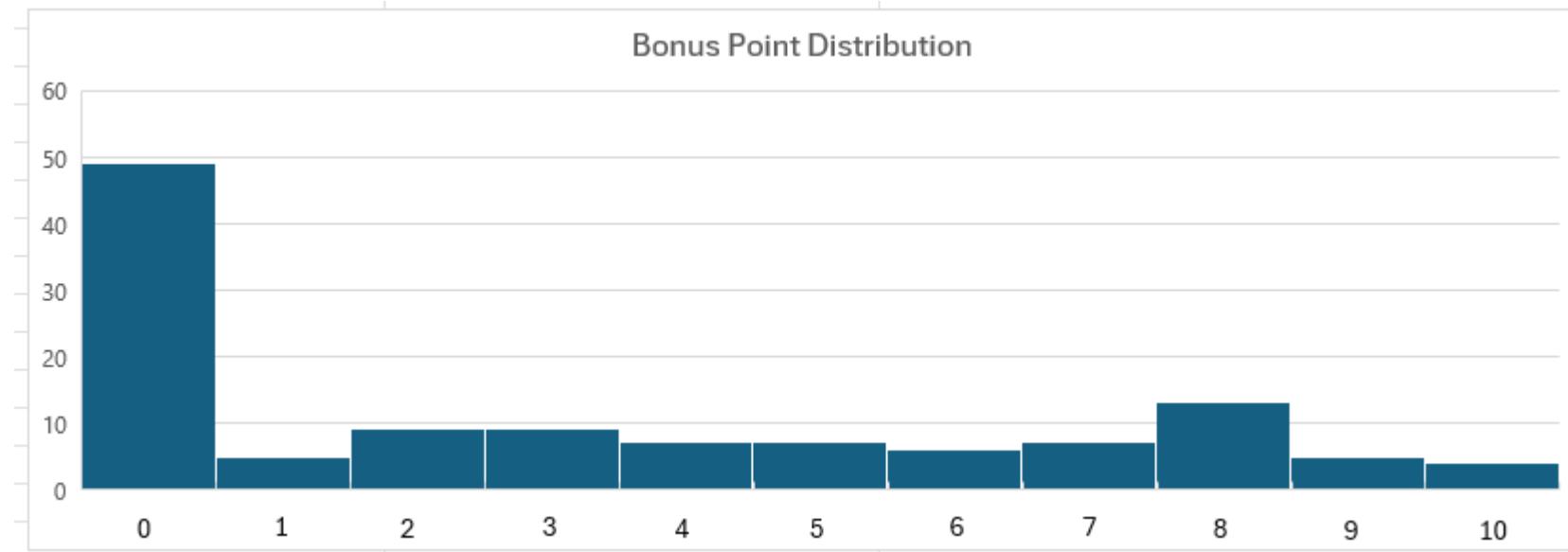


Online resources website (same for global edition and the North-America one):  
Includes applets/ animations, blogg, interactive excrcises, ...  
[https://media.pearsoncmg.com/ph/esm/cs\\_kurose\\_compnetwork\\_8/cw](https://media.pearsoncmg.com/ph/esm/cs_kurose_compnetwork_8/cw)

# Bonuspoint Program (Beta!)

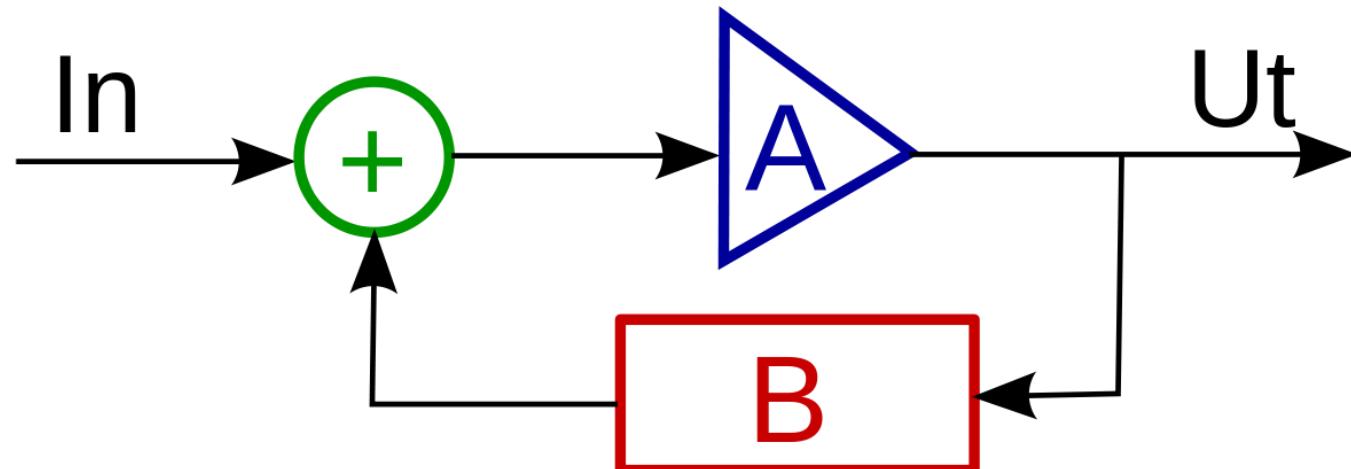


- We are testing a new bonuspoint program this year.
- At random occasions (lectures or exercises) we will conduct a "surprise" quiz in class.
- In each quiz you could earn up to 2 bonus points.
- **Once you reached 10 points, you cannot earn more extra points.**
- **The bonus points can only be used once and only in the exam in March 2024 which has a total of 60 points!**



# Provide your feedback in the course evaluation!!

- It is so important to have your feedback (both positive and negative).
- Without your feedback, we cannot improve our courses in a good way.
- We tested new elements (Path to Pass, Bonus Point System). We need your feedback to know if they were helpful for you.
- **You will receive a link to the course survey Monday after the exam.**



# Roadmap



- Recalling from introduction
  - A day in the life of a web request
- Some random repetition
  - UDP Checksum, Error Detection
  - Classless Inter-Domain Routing
  - Dijkstra's algorithm
  - CSMA
  - CDMA
- Reflections, Perspectives

# Layers, Protocols, Interfaces

Each **layer implements services**

- via its own internal-layer actions
- and via ‘the services by layer below

Each layer **provides** services to the upper layer  
(shielding from implementation details) and **uses**  
services of the layer below

Layer  $n$  on a host carries a conversation with  
layer  $n$  on another host

**host-to-host interface**: defines message exchanges (incl.  
format, order, and more) with peer entity = **logical  
communication, protocol**

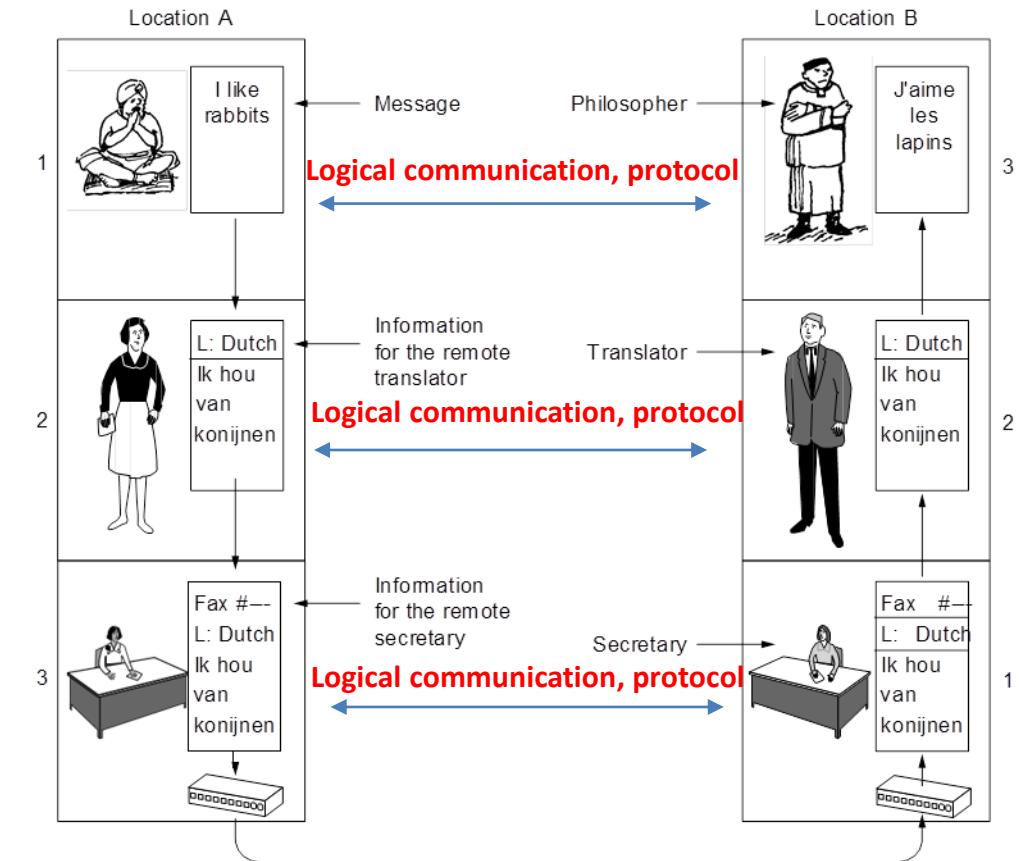


Fig. 1-10. The philosopher-translator-secretary architecture.

Fig. A. Tanenbaum Computer Networks

# The OSI Reference Model

ISO (International Standards Organization) defined the OSI (Open Systems Interconnect) model to help vendors create interoperable network implementation

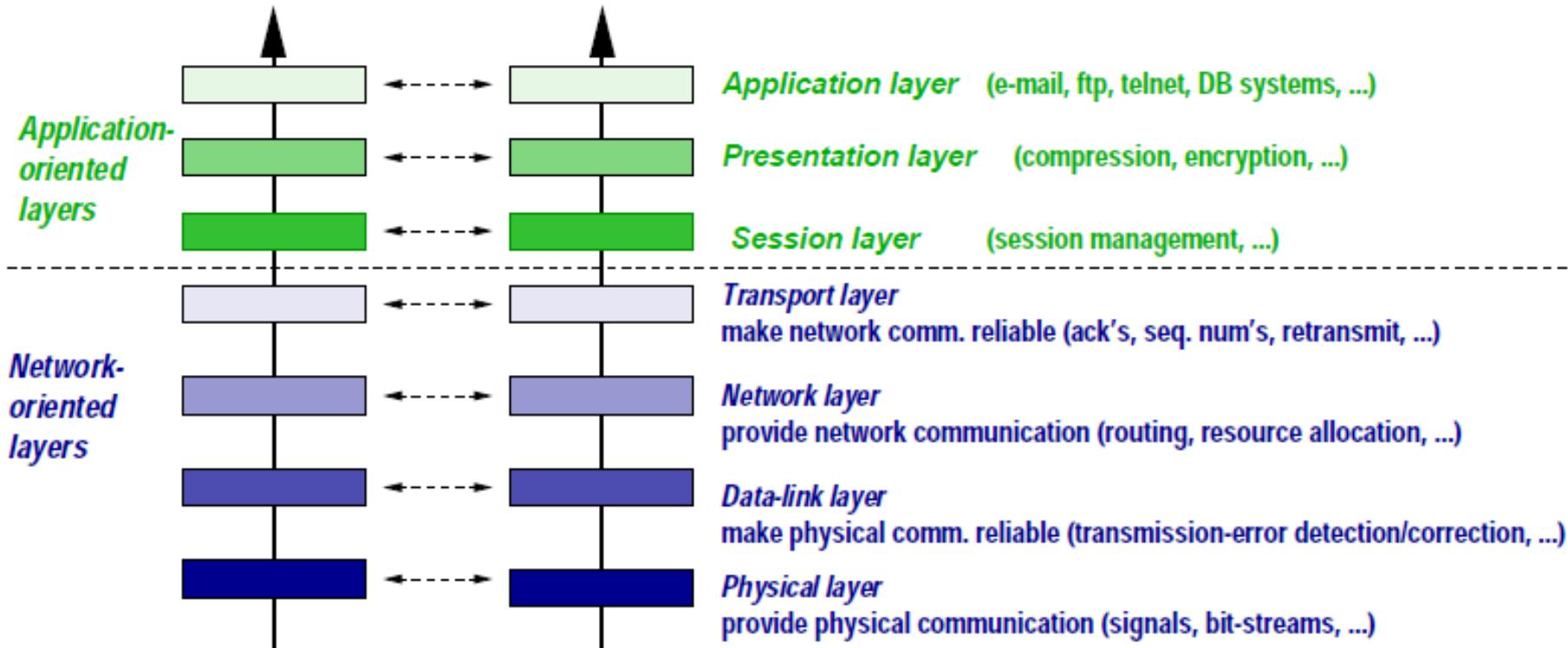


Fig. Steen, Sips : Computer and Network organization

``X dot'' series (X.25, X.400, X.500) OSI model implementation (protocol stack)

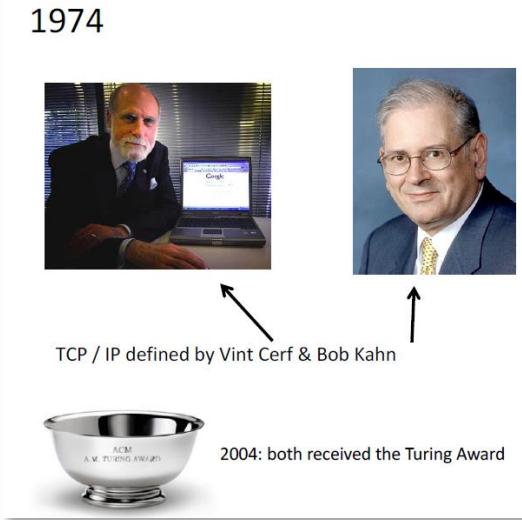
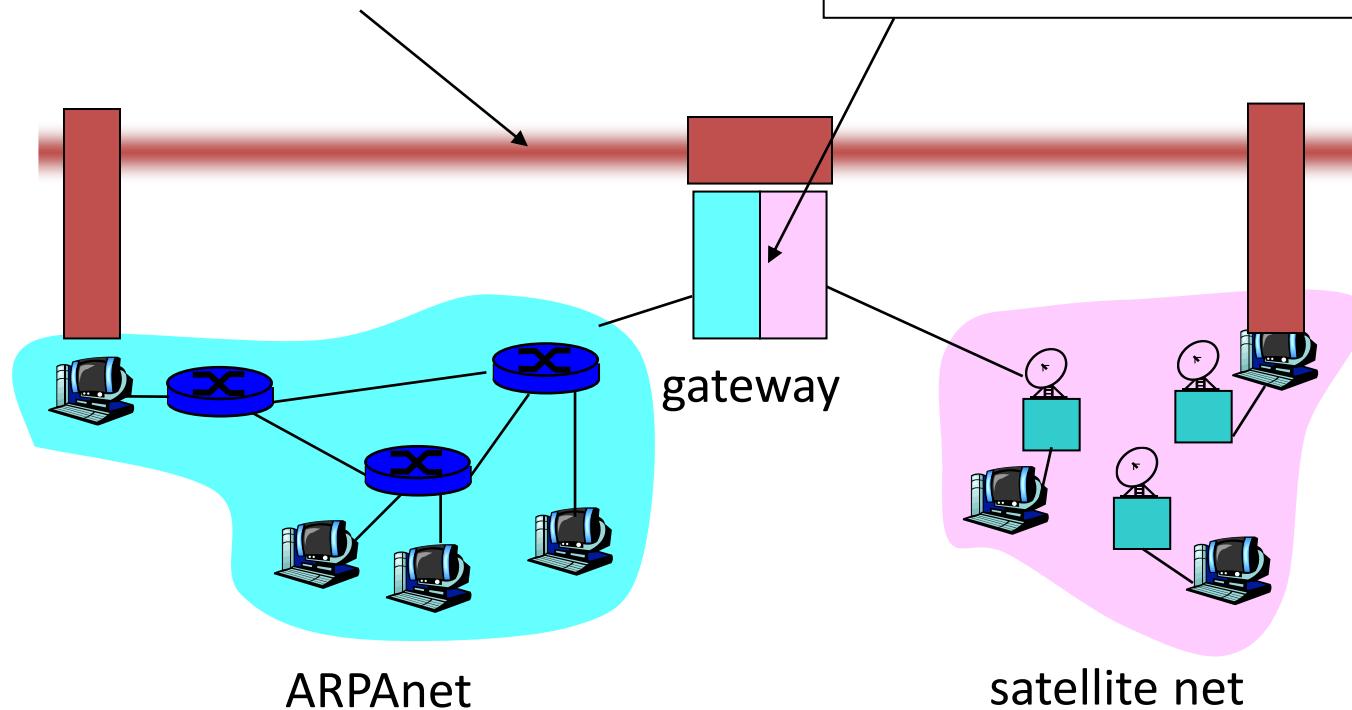
# The Internet: bridging networks

Internet layer (IP):

- internetwork appears as single, uniform entity, despite underlying local network heterogeneity
- network of networks

Gateway:

- “embed internetwork packets in local packet format”
- route (at inter-network level) to next gateway

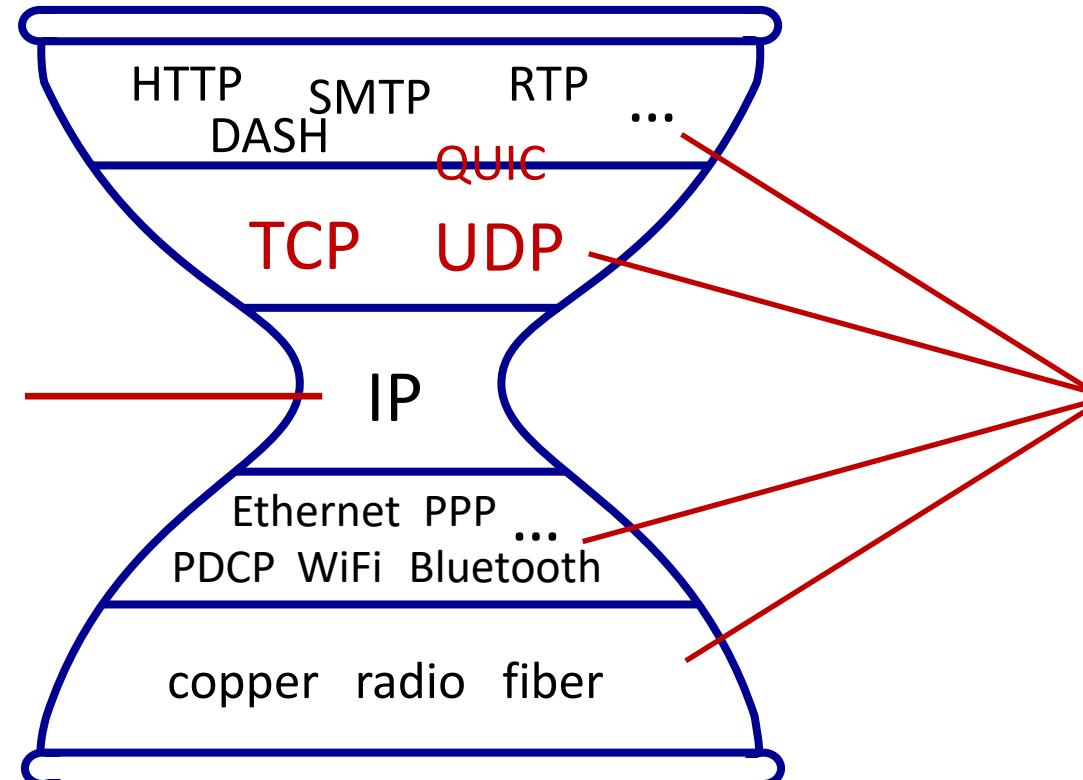


"A Protocol for Packet Network Intercommunication", V. Cerf, R. Kahn, IEEE Transactions on Communications, May, 1974, pp. 637-648.

# Internet protocol to rule them all...

Internet's "thin waist":

- *one* network layer protocol: IP
- *must* be implemented by every (billions) of Internet-connected devices



Read also: On the hourglass model, CACM 2019  
[https://www.youtube.com/watch?v=L9s096\\_r\\_U](https://www.youtube.com/watch?v=L9s096_r_U)

*many* protocols in application, transport, link and physical layers

Used in the design of the Internet and Unix, the layered services of the hourglass model have enabled viral adoption and deployment scalability.  
BY MICAH BECK

On The Hourglass Model

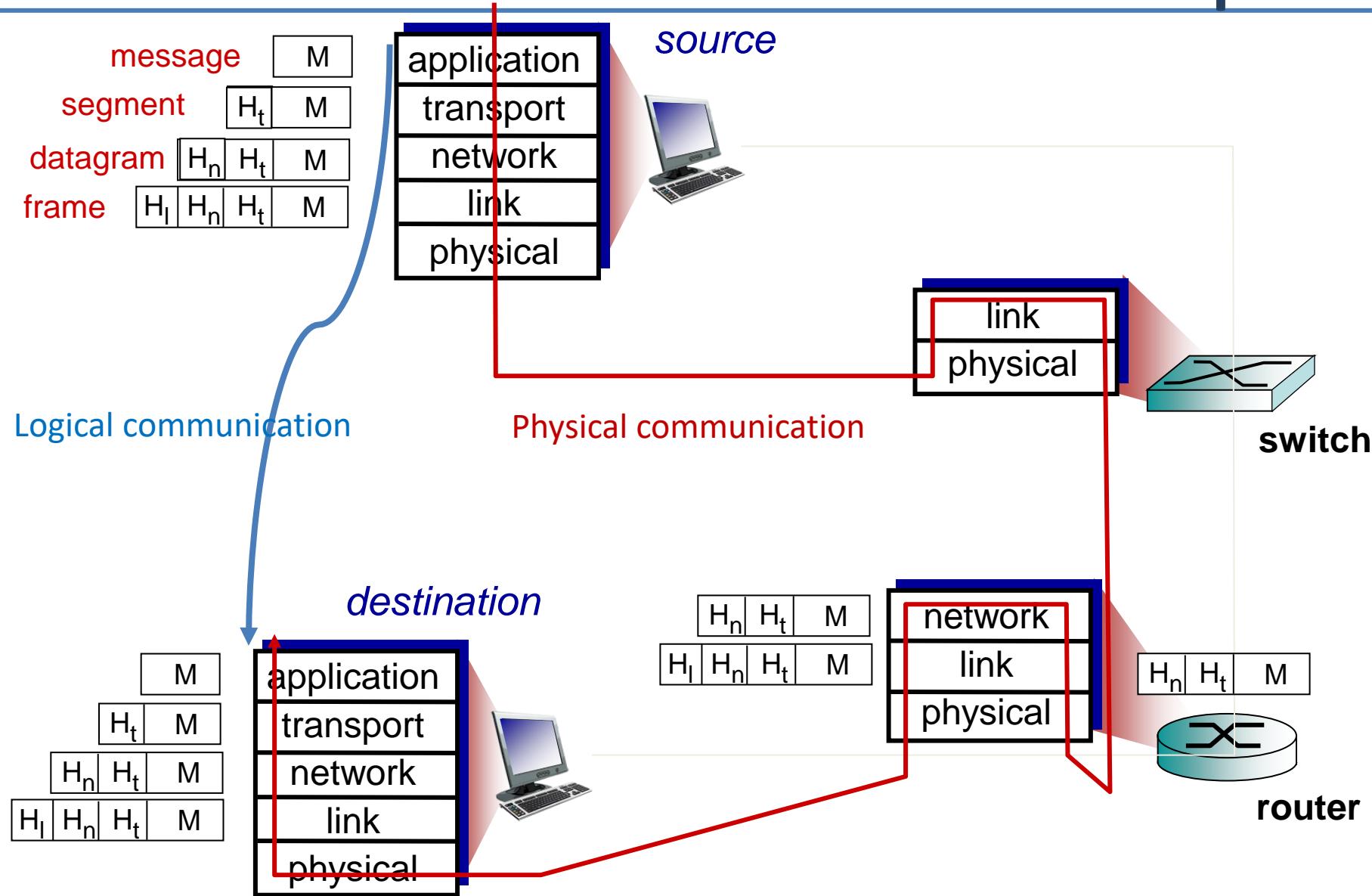
THE HOURGLASS MODEL of layered systems architecture is a visual and conceptual representation of an approach to design that seeks to support a great

layer by it tailored diff technologies in above" an tocal as th net (see F ning layer of the Int access and DHCP)

The sh glass mo the span various a mtable poros h glass as the intu functions is instr goals. Th are to an hour "waist" of the restri large upp ing the i and supp

The ho

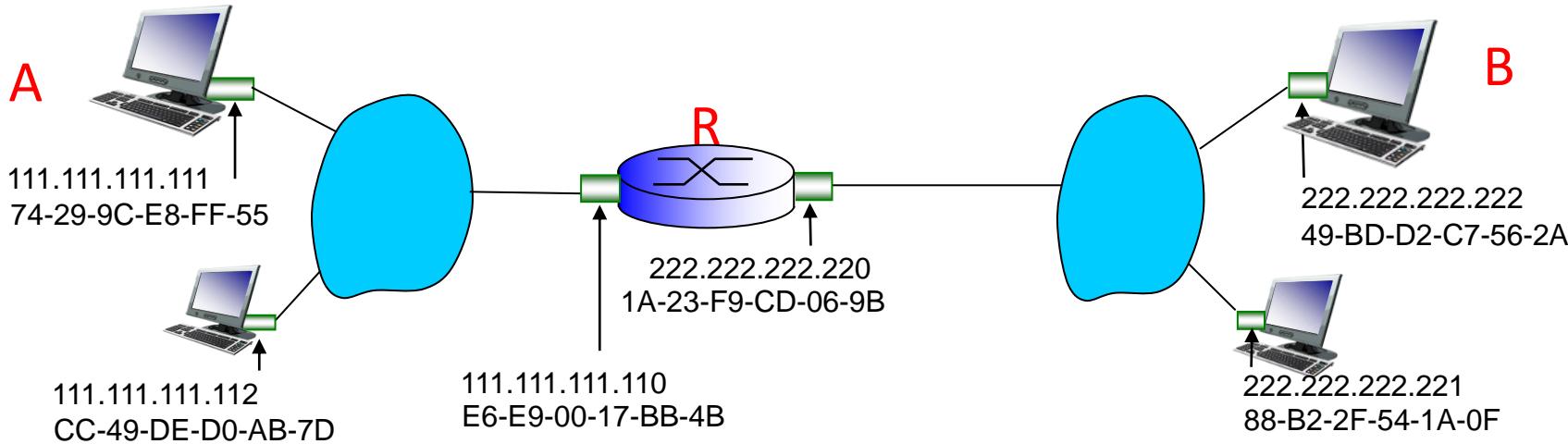
# (recall) Layered communication: Encapsulation



# Addressing: routing to another LAN

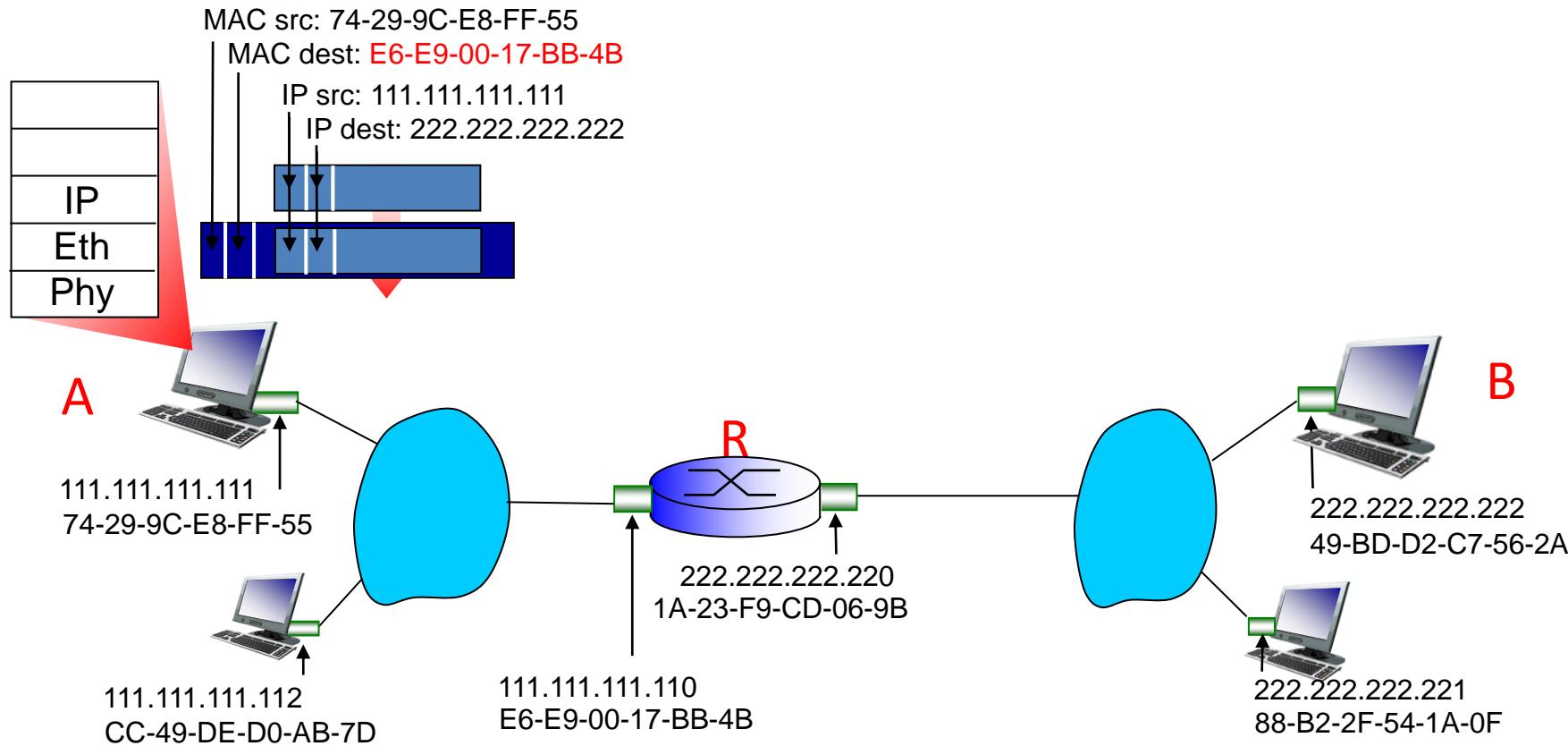
walkthrough: send datagram from A to B via R

- focus on addressing – at IP (datagram) and MAC layer (frame)
- assume A knows B's IP address (how?)
- assume A knows IP address of first hop router, R (how?)
- assume A knows R's MAC address (how?)



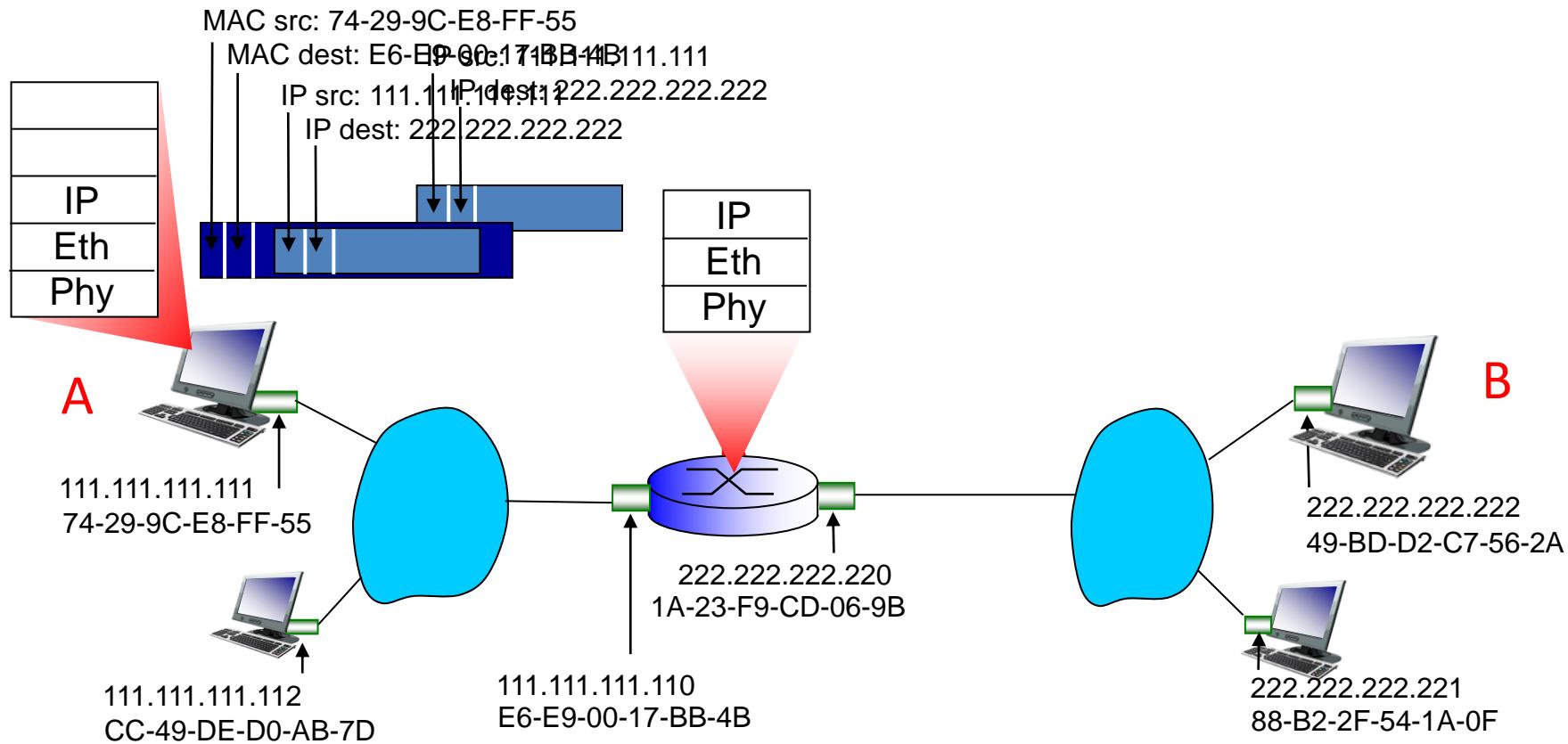
# Addressing: routing to another LAN

- A creates IP datagram with IP source A, destination B
- A creates link-layer frame with R's MAC address as destination address, frame contains A-to-B IP datagram



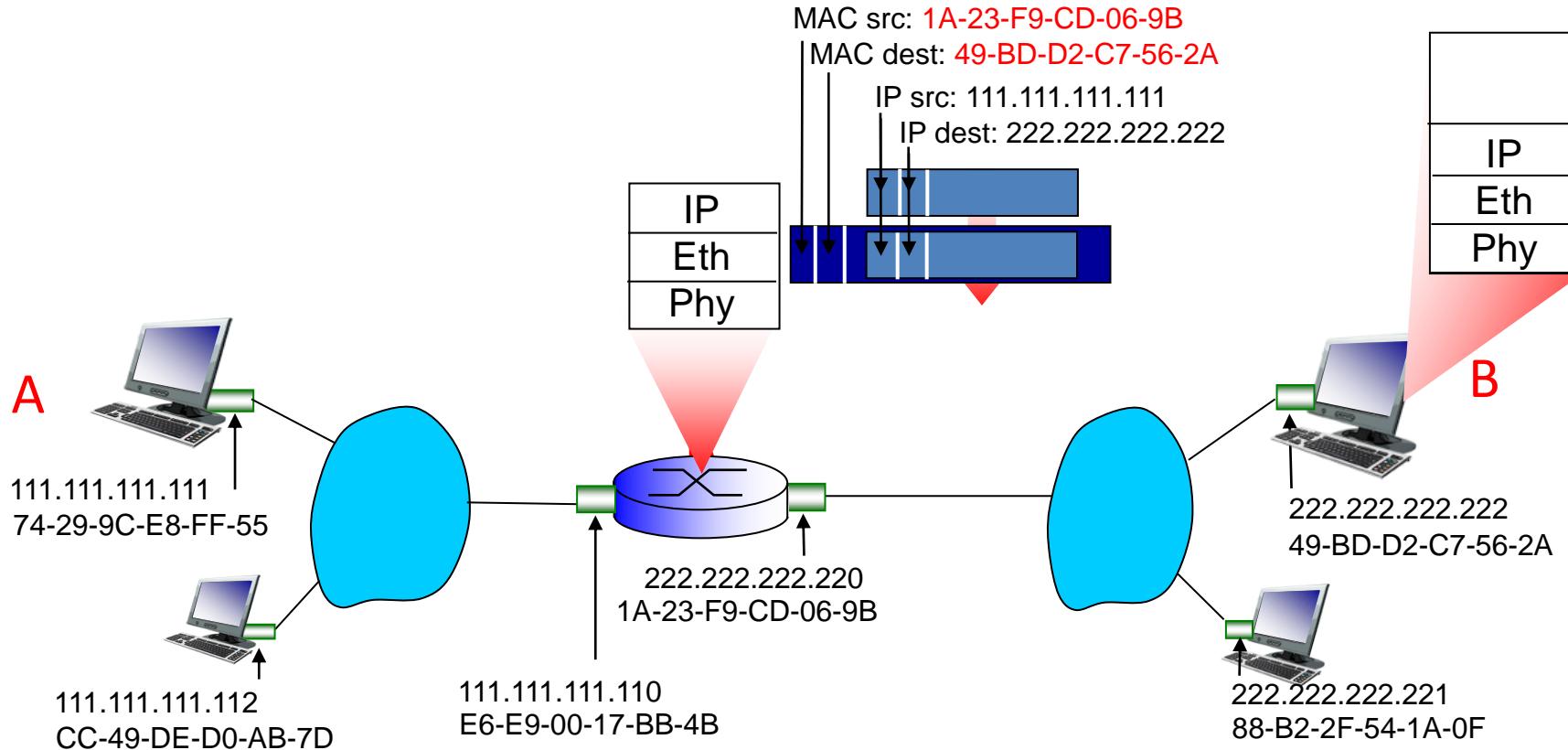
# Addressing: routing to another LAN

- frame sent from A to R
- frame received at R, datagram removed, passed up to IP



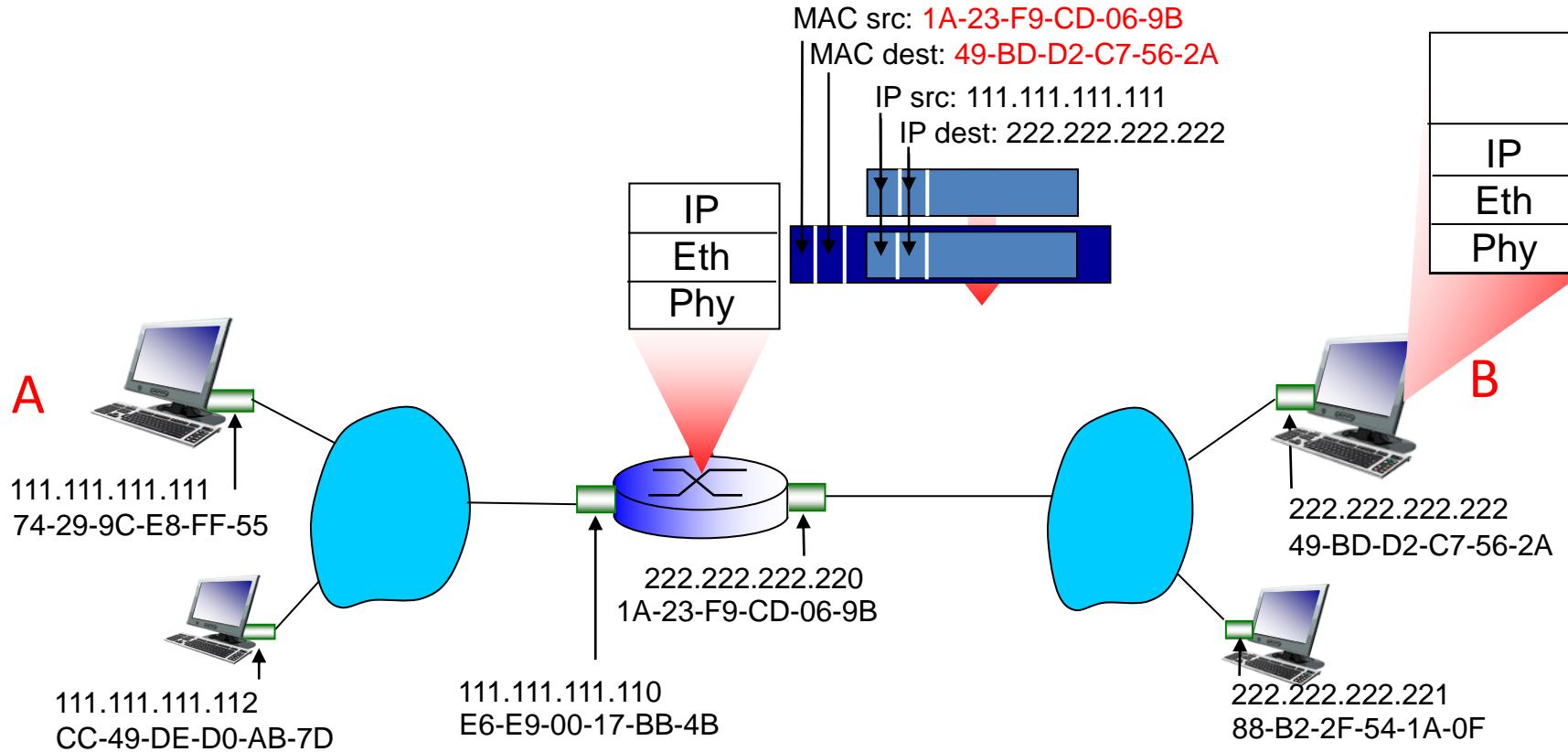
# Addressing: routing to another LAN

- R forwards datagram with IP source A, destination B
- R creates link-layer frame with B's MAC address as destination address, frame contains A-to-B IP datagram



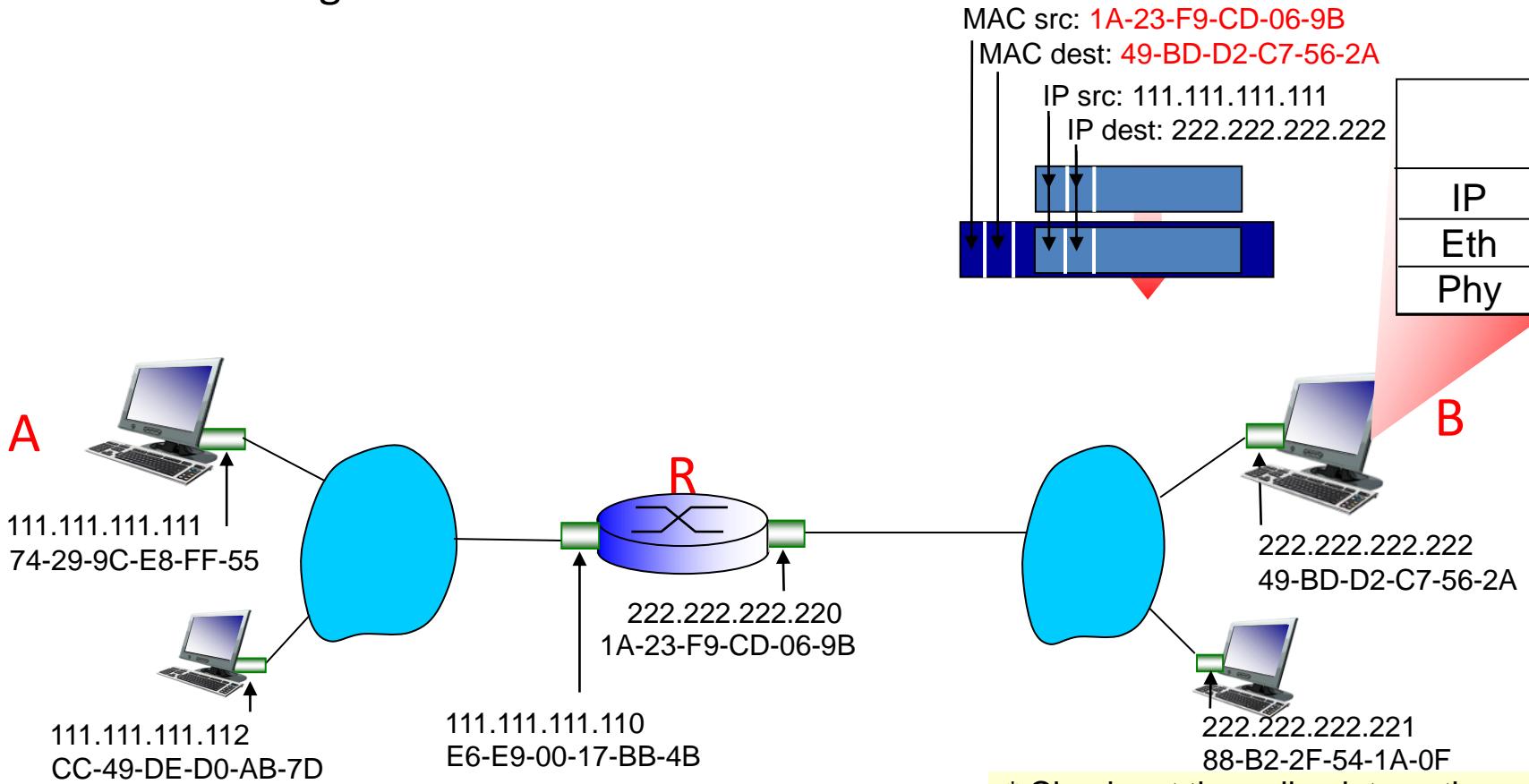
# Addressing: routing to another LAN

- R forwards datagram with IP source A, destination B
- R creates link-layer frame with B's MAC address as destination address, frame contains A-to-B IP datagram



# Addressing: routing to another LAN

- R forwards datagram with IP source A, destination B
- R creates link-layer frame with B's MAC address as dest, frame contains A-to-B IP datagram



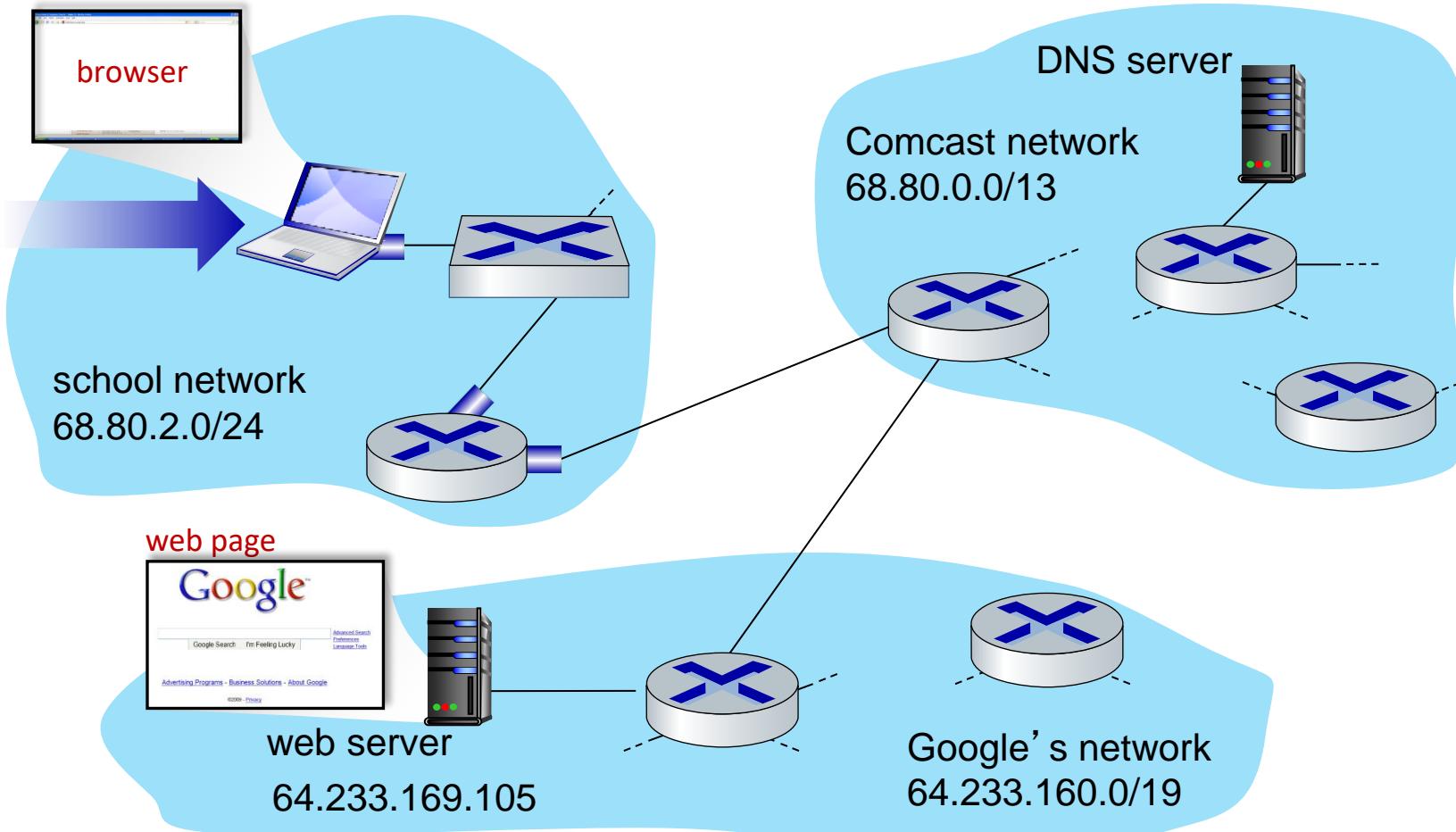
\* Check out the online interactive exercises for more examples: [http://gaia.cs.umass.edu/kurose\\_ross/interactive/](http://gaia.cs.umass.edu/kurose_ross/interactive/)

# Roadmap



- Recalling from introduction
  - A day in the life of a web request
- Some random repetition
  - UDP Checksum, Error Detection
  - Classless Inter-Domain Routing
  - Dijkstra's algorithm
  - CSMA
  - CDMA
- Reflections, Perspectives

# A day in the life of a web request: The scenario

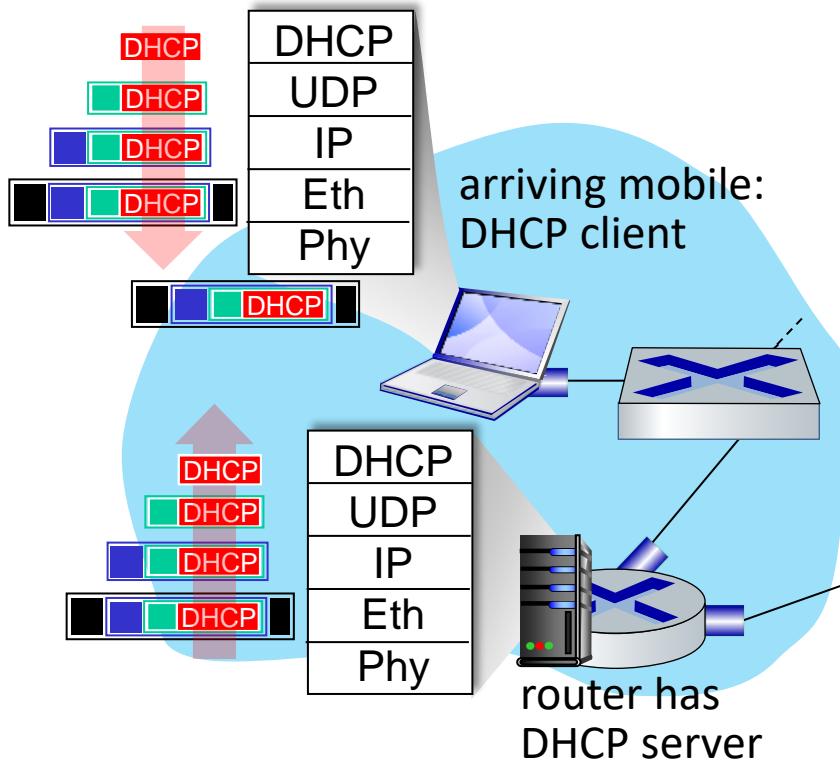


scenario:

- arriving mobile client attaches to network ...
- requests web page:  
www.google.com

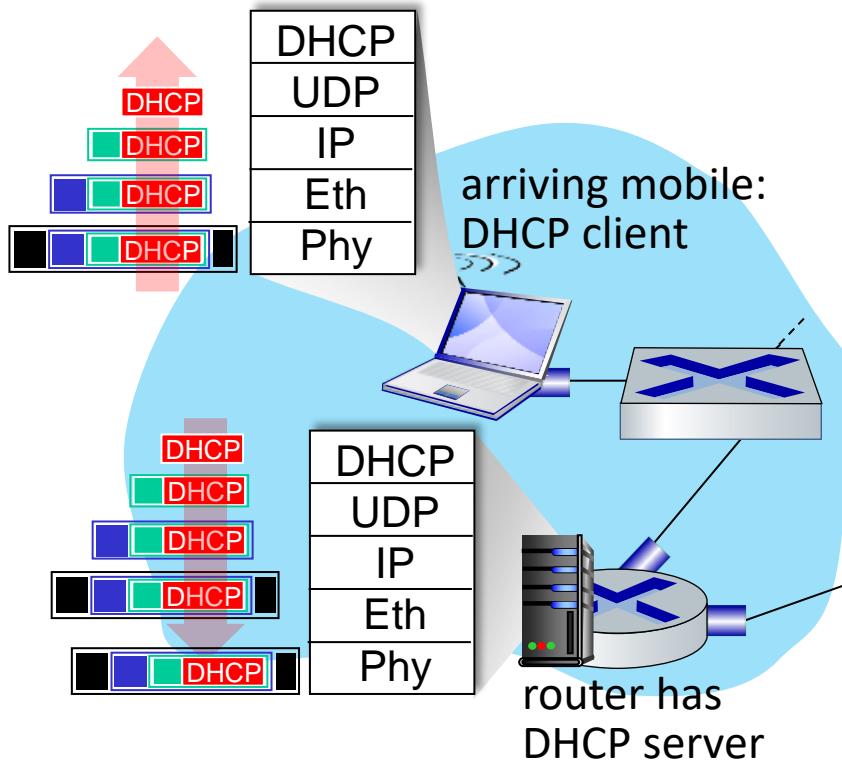
*Sounds simple!* !

# A day in the life: connecting to the Internet



- connecting laptop needs to get its own IP address, addr of first-hop router, addr of DNS server: use **DHCP**
- DHCP request encapsulated in **UDP**, encapsulated in **IP**, encapsulated in **802.3** Ethernet frame
- Ethernet frame **broadcast** (dest: FFFFFFFFFFFF) on LAN, received at router running **DHCP** server
- Ethernet **demuxed** to IP, demuxed to UDP, demuxed to DHCP

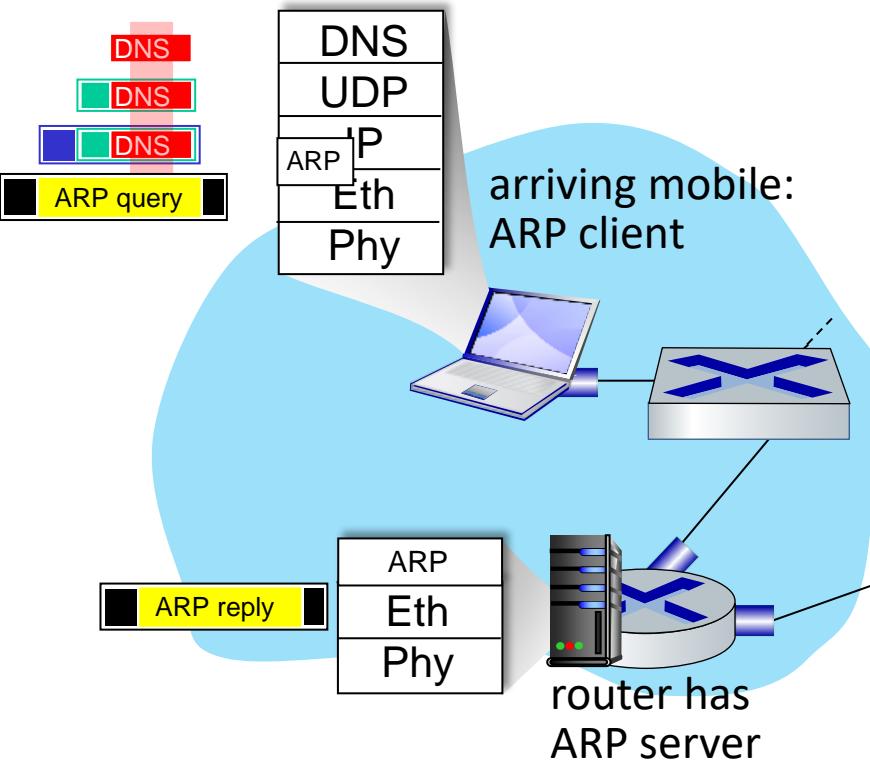
# A day in the life: connecting to the Internet



- DHCP server formulates **DHCP ACK** containing client's IP address, IP address of first-hop router for client, name & IP address of DNS server
- encapsulation at DHCP server, frame forwarded (**switch learning**) through LAN, demultiplexing at client
- DHCP client receives DHCP ACK reply

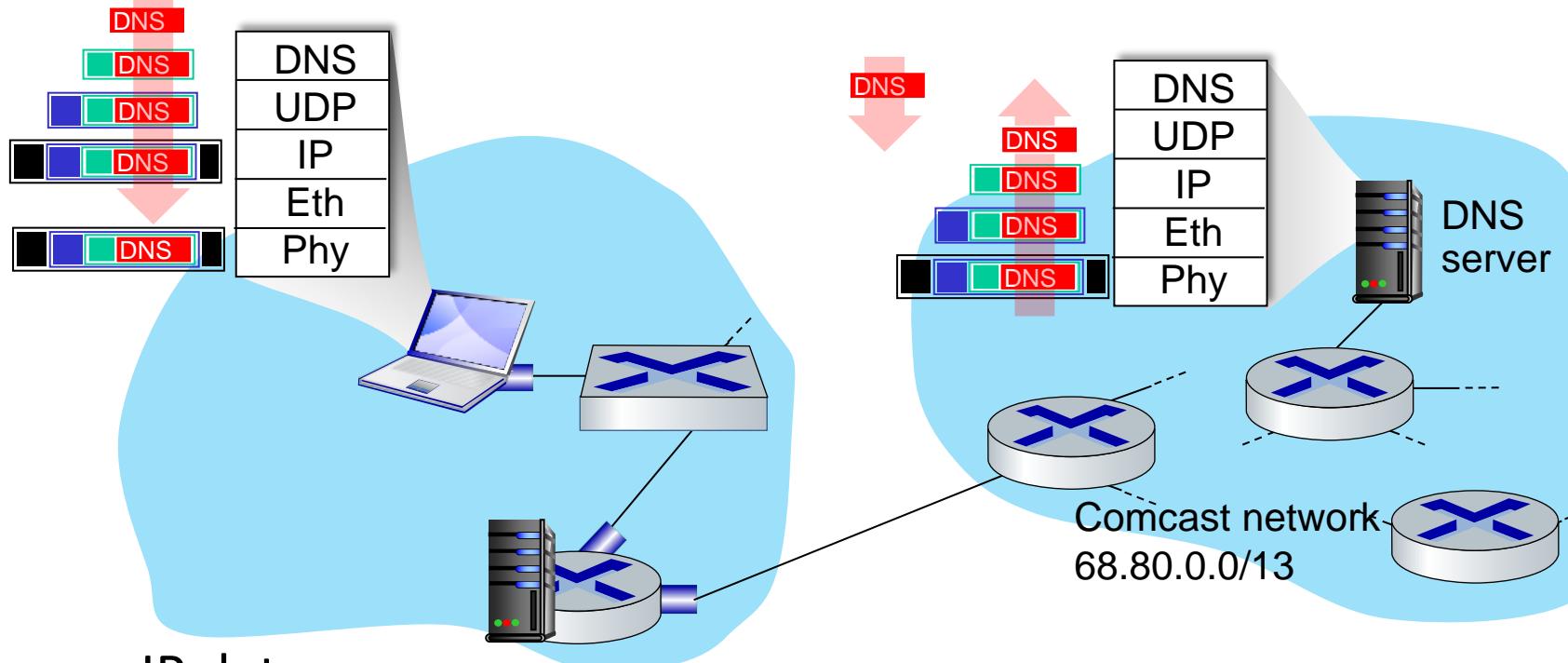
*Client now has IP address, knows name & addr of DNS server, IP address of its first-hop router*

# A day in the life... ARP (before DNS, before HTTP)



- before sending **HTTP** request, need IP address of www.google.com: **DNS**
- DNS query created, encapsulated in UDP, encapsulated in IP, encapsulated in Eth. To send frame to router, need MAC address of router interface: **ARP**
- **ARP query** broadcast, received by router, which replies with **ARP reply** giving MAC address of router interface
- client now knows MAC address of first hop router, so can now send frame containing DNS query

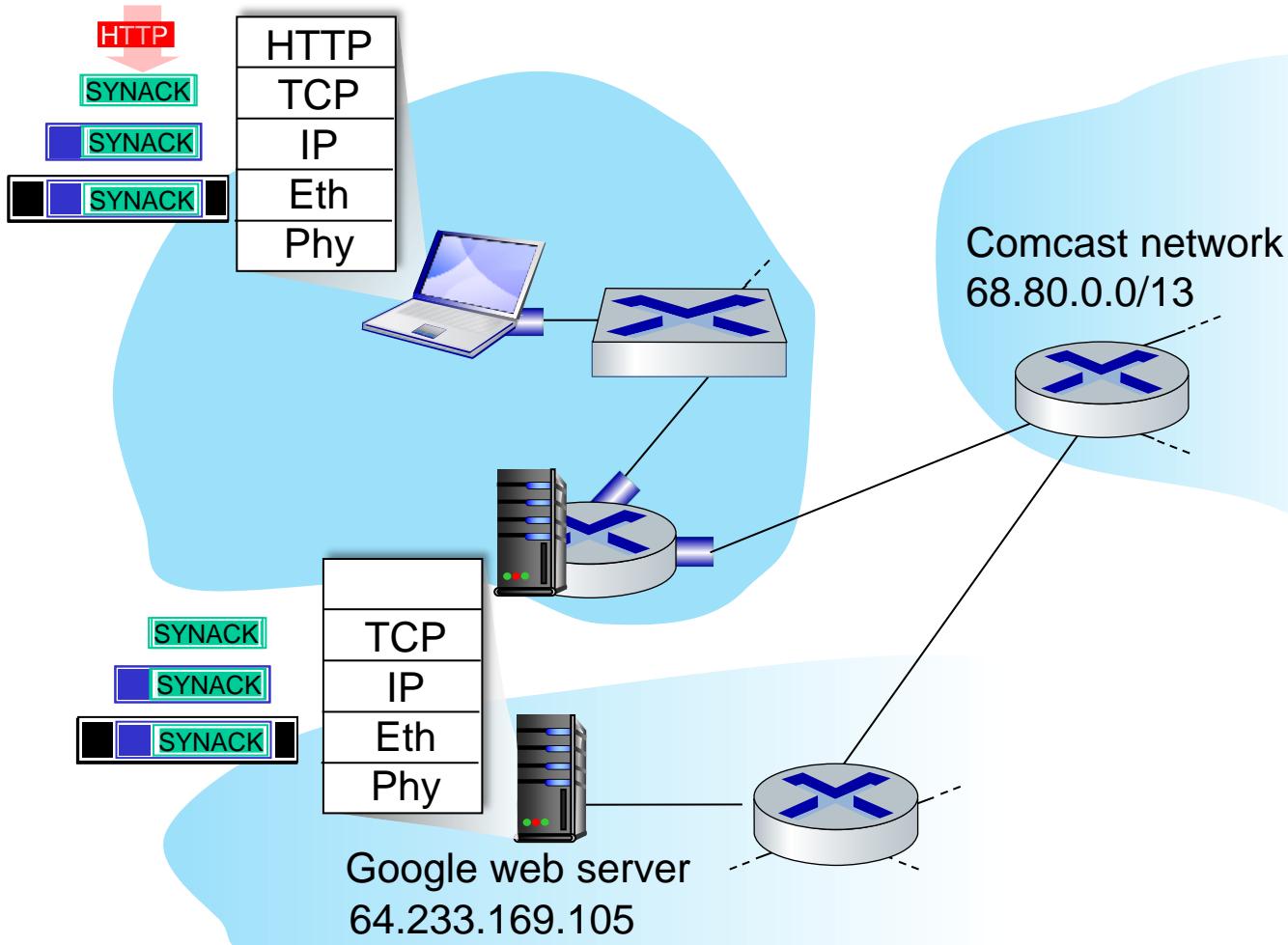
# A day in the life... using DNS



- IP datagram containing DNS query forwarded via LAN switch from client to 1<sup>st</sup> hop router
- IP datagram forwarded from campus network into Comcast network, routed (tables created by **RIP, OSPF, IS-IS** and/or **BGP** routing protocols) to DNS server

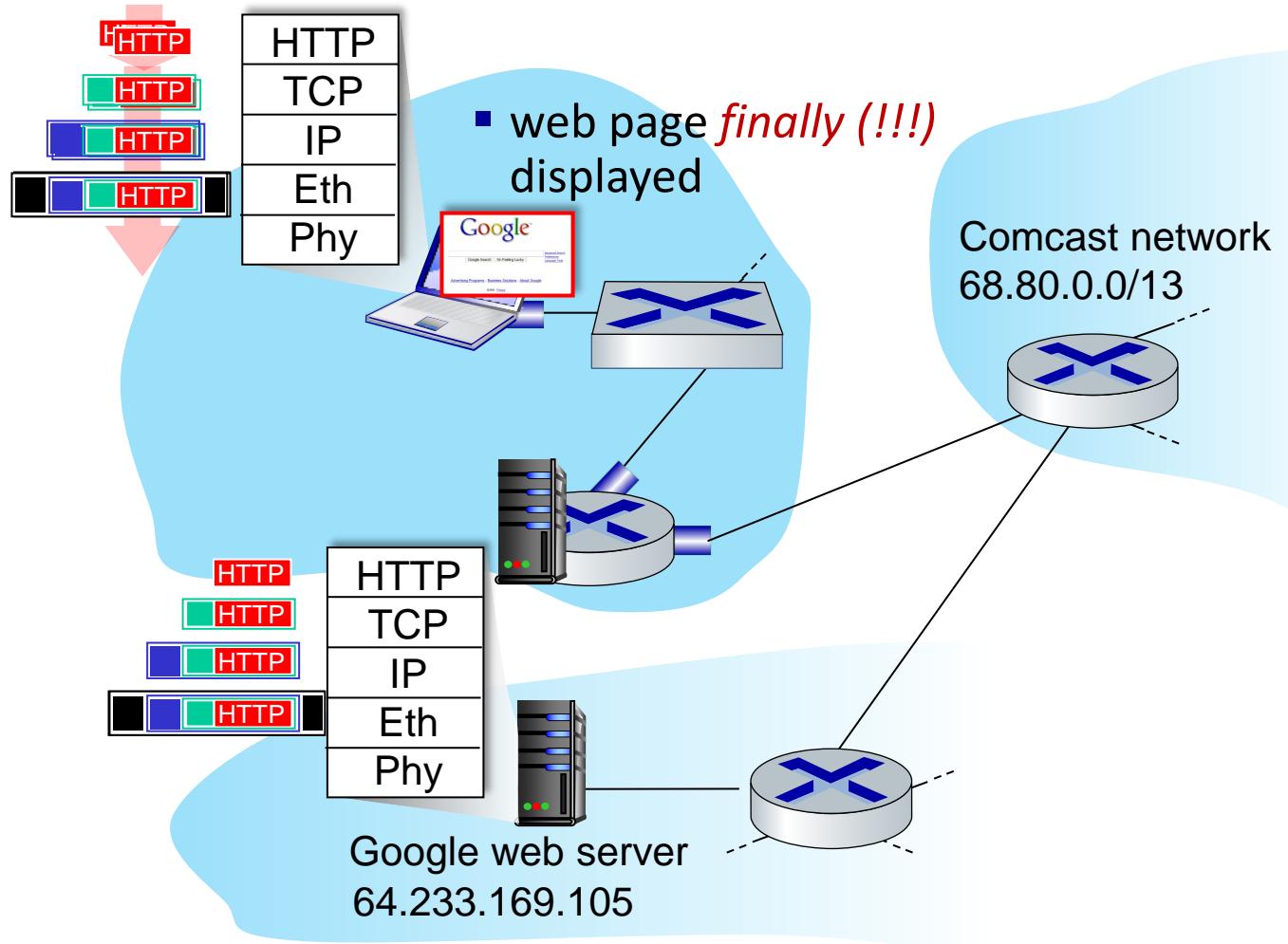
- demuxed to DNS
- DNS replies to client with IP address of [www.google.com](http://www.google.com)

# A day in the life...TCP connection carrying HTTP



- to send HTTP request, client first opens **TCP socket** to web server
- **TCP SYN segment** (step 1 in TCP 3-way handshake) inter-domain routed to web server
- web server responds with **TCP SYNACK** (step 2 in TCP 3-way handshake)
- **TCP connection established!**

# A day in the life... HTTP request/reply



- **HTTP request sent into TCP socket**
- **IP datagram containing HTTP request routed to www.google.com**
- **web server responds with HTTP reply (containing web page)**
- **IP datagram containing HTTP reply routed back to client**

# Roadmap



- Recalling from introduction
  - A day in the life of a web request
- Some random repetition
  - UDP Checksum, Error Detection
  - Classless Inter-Domain Routing
  - Dijkstra's algorithm
  - CSMA
  - CDMA
- Reflections, Perspectives

# Request 1: UDP Checksum

---

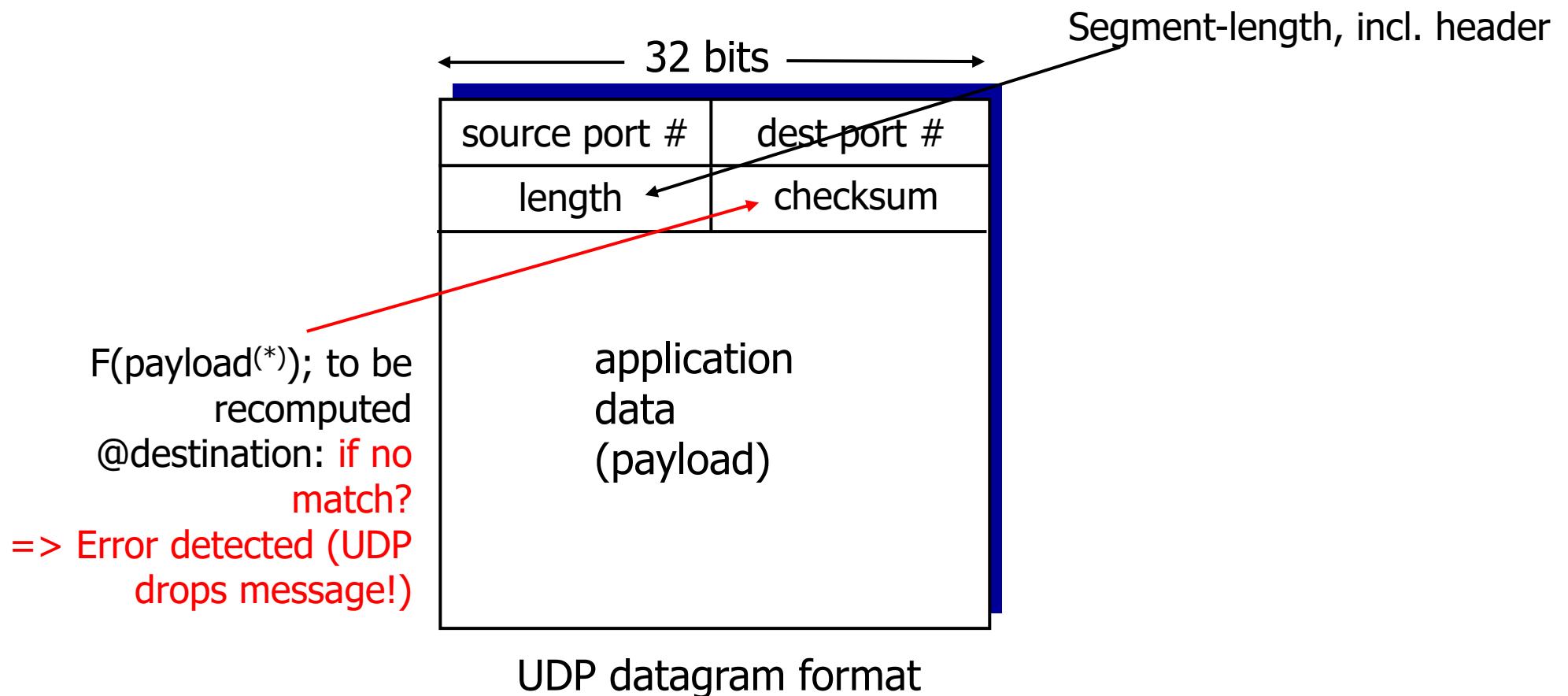
Could you go through a couple of examples of UDP checksum algorithm and CDR.

## Request 2: Error Detection

---

I would like you to go through subjects of  
**error detection.**

# UDP: segment header



(\*) addition (1's complement sum) of contents,  
seen as seq. of 16-bit numbers

# Internet checksum

---

**Goal:** detect errors (*i.e.*, flipped bits) in transmitted segment

## Sender:

- treat contents of UDP segment (including UDP header fields and IP addresses) as sequence of 16-bit integers
- **checksum:** addition (one's complement sum) of segment content
- checksum value put into UDP checksum field

## Receiver:

- compute checksum of received segment
- check if computed checksum equals checksum field value:
  - not equal - error detected
  - equal - no error detected. **But maybe errors nonetheless?** More later ....

# UDP Checksum [RFC 1071]

## Sender:

- treat segment contents as sequence of 16-bit integers
- checksum: addition (1's complement sum) of segment contents
- sender puts checksum value into UDP checksum field

## Receiver:

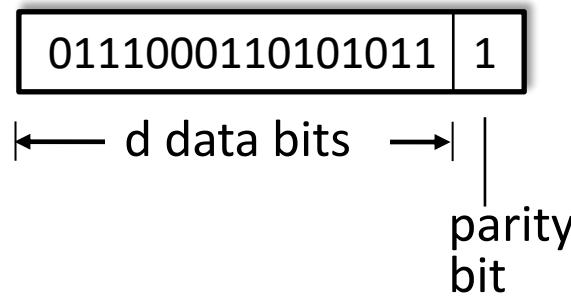
- compute checksum of received segment
- check if **computed checksum == checksum field** value:
  - NO - error detected (*report error to app or discard*)
  - YES - no error detected.
    - *But maybe (rarely) errors nonetheless?* More later ....

|                             |                                                                                      |
|-----------------------------|--------------------------------------------------------------------------------------|
|                             | 1 1 1 0 0 1 1 0 0 1 1 0 0 1 1 0                                                      |
|                             | 1 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1                                                      |
| <hr/>                       |                                                                                      |
| Wraparound:<br>Add to final |  |
| sum                         | 1 0 1 1 1 0 1 1 1 0 1 1 1 1 0 0                                                      |
| checksum                    | 0 1 0 0 0 1 0 0 0 1 0 0 0 0 1 1                                                      |

# Parity checking

## Single bit parity:

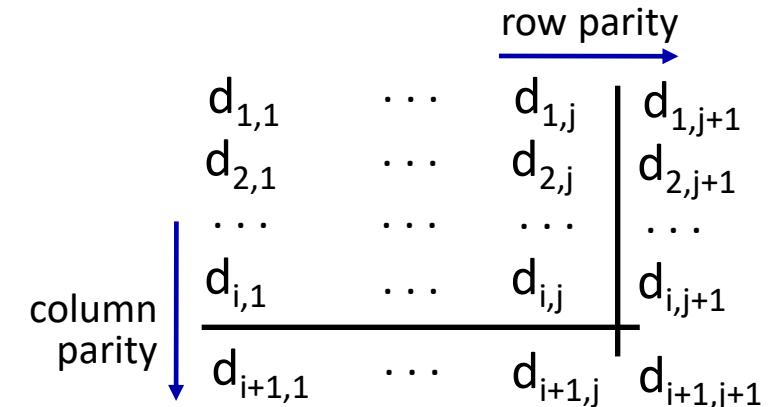
- detect single bit errors



Even parity: set parity bit so there is an even number of 1's

## Two-dimensional bit parity:

- detect *and correct* single bit errors



no errors: 
$$\begin{array}{c|c} 1 & 0 & 1 & 0 & 1 & | & 1 \\ 1 & 1 & 1 & 1 & 0 & | & 0 \\ 0 & 1 & 1 & 1 & 0 & | & 1 \\ \hline 0 & 0 & 1 & 0 & 1 & | & 0 \end{array}$$

detected  
and  
correctable  
single-bit  
error:

A binary sequence  $10101100$  is shown in a box. A red arrow labeled "parity error" points to the fourth bit from the left. A red arrow labeled "parity error" also points to the last bit.

# Roadmap

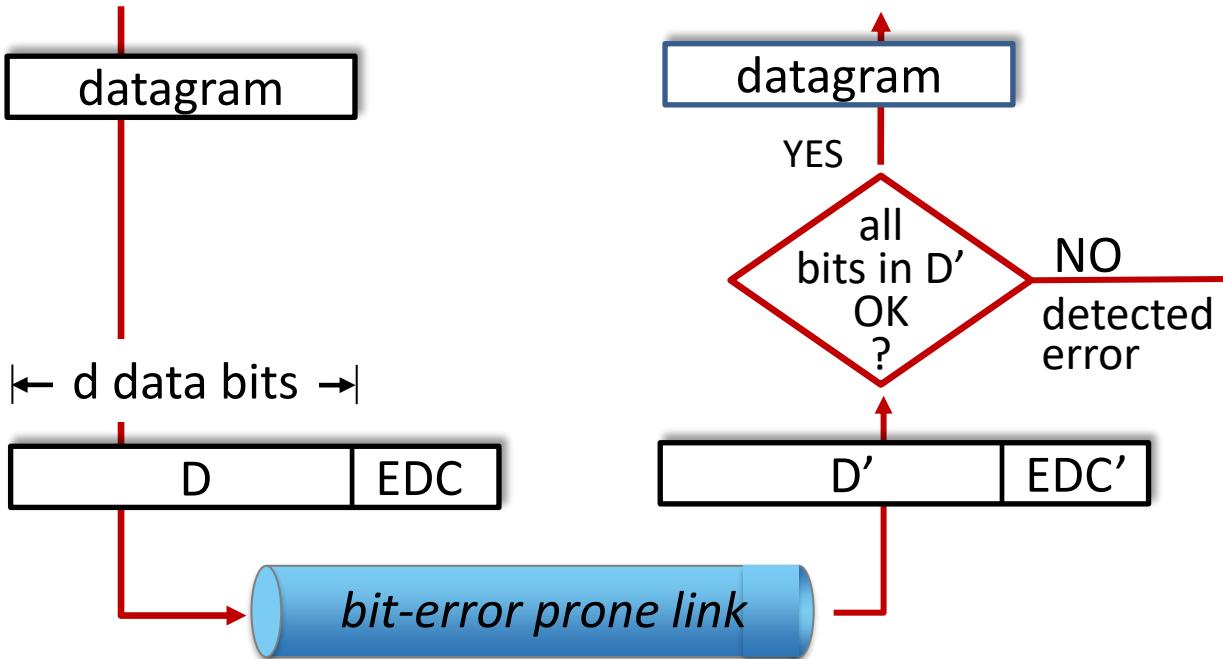


- Recalling from introduction
  - A day in the life of a web request
- Some random repetition
  - UDP Checksum, Error Detection
  - Classless Inter-Domain Routing
  - Dijkstra's algorithm
  - CSMA
  - CDMA
- Reflections, Perspectives

# Error detection

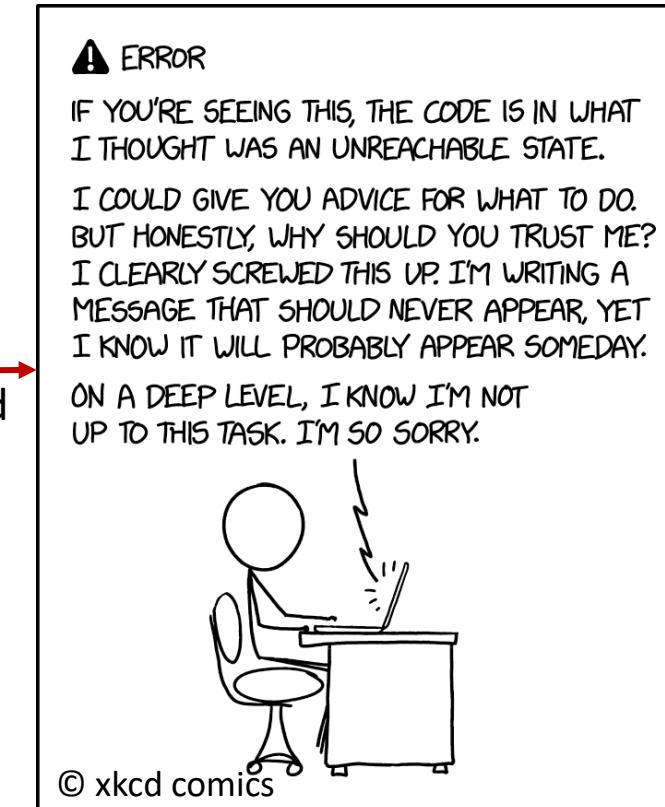
**EDC:** Error detection and correction bits (e.g., redundancy)

**D:** Data protected by error checking, may include header fields



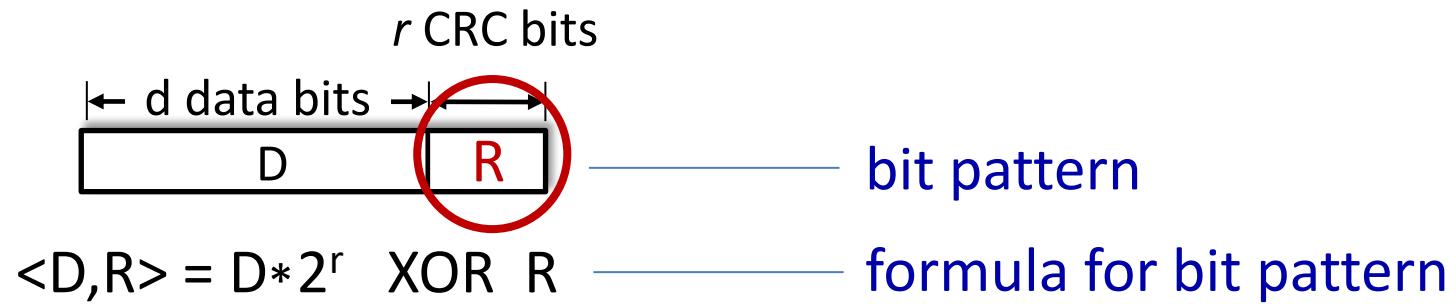
Error detection not 100% reliable!

- protocol may miss some errors, but rarely
- larger EDC field yields better detection and correction



# Cyclic Redundancy Check (CRC)

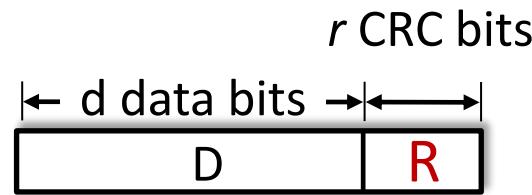
- More powerful error-detection coding
- **D**: data bits (given, think of these as a binary number)
- **G**: bit pattern (generator), of  $r+1$  bits (given)



Goal: choose  $r$  CRC bits, **R**, such that  $\langle D, R \rangle$  exactly divisible by  $G$  ( $\text{mod } 2$ )

- Receiver knows  $G$ , divides  $\langle D, R \rangle$  by  $G$ . If non-zero remainder: error detected!
- Can detect all burst errors less than  $r+1$  bits
- Widely used in practice (Ethernet, 802.11 WiFi)

# Cyclic Redundancy Check (CRC)



$$D \cdot 2^r \text{XOR } R = nG$$

$$D \cdot 2^r = nG \text{ XOR } R$$

$$\frac{D \cdot 2^r}{G} = n \text{ XOR } R$$

$$R = \text{remain.} \frac{D \cdot 2^r}{G}$$

- We choose  $R$  such that  $G$  divides into  $D \cdot 2^r$  XOR  $R$  without remainder.
- XOR  $R$  on both side of the eq.
- Divide both sides by  $G$
- What is left in that division is our desired  $R$ .

$D=10011110$ ,  $r=3$ ,  
 $G=1001$  ( $r+1$  bit)

|              |              |
|--------------|--------------|
| 10011110 000 | 10011110 111 |
| 1001         | 1001         |
| 00001110 000 | 00001110 111 |
| 1001         | 1001         |
| 00000111 000 | 00000111 111 |
| 100 1        | 100 1        |
| 00000011 100 | 00000011 011 |
| 10 01        | 10 01        |
| 00000001 110 | 00000001 001 |
| 1 001        | 1 001        |
| 00000000 111 | 00000000 000 |

# Another Example

---

10011001 000

1001

00001001 000

1001

00000000 000

## Request 2: Smallest CIDR

---

*I would like you to go through subjects of Smallest CIDR.*

# The OSI Reference Model

ISO (International Standards Organization) defined the OSI (Open Systems Interconnect) model to help vendors create interoperable network implementation

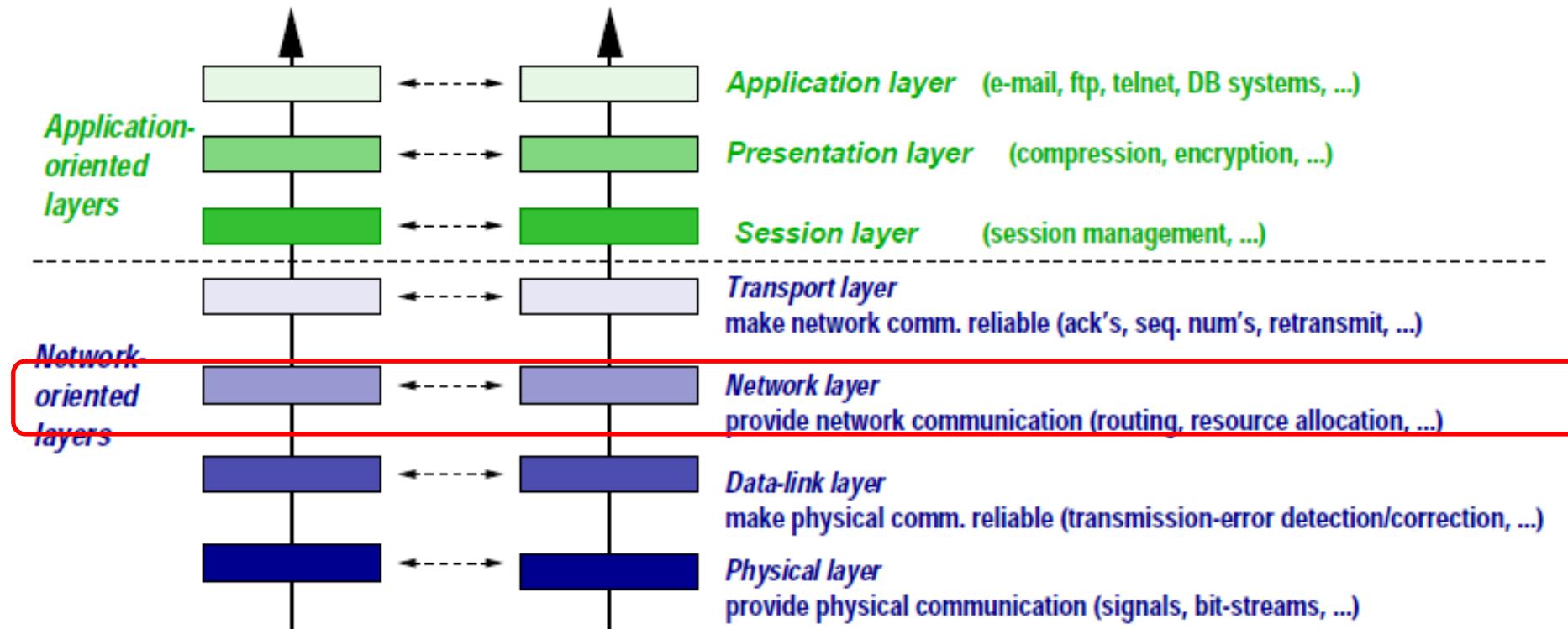
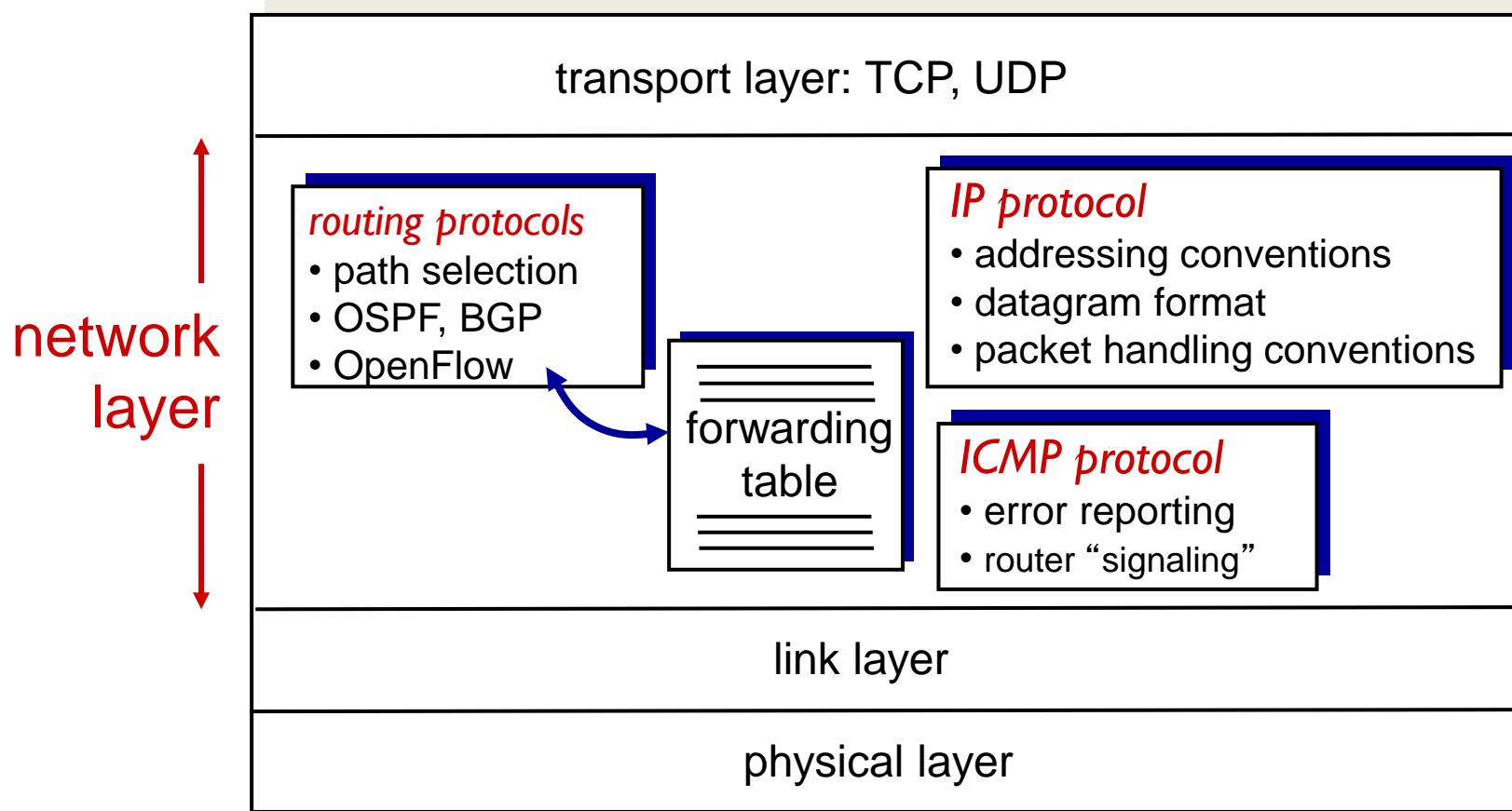


Fig. Steen, Sips : Computer and Network organization

``X dot'' series (X.25, X.400, X.500) OSI model implementation (protocol stack)

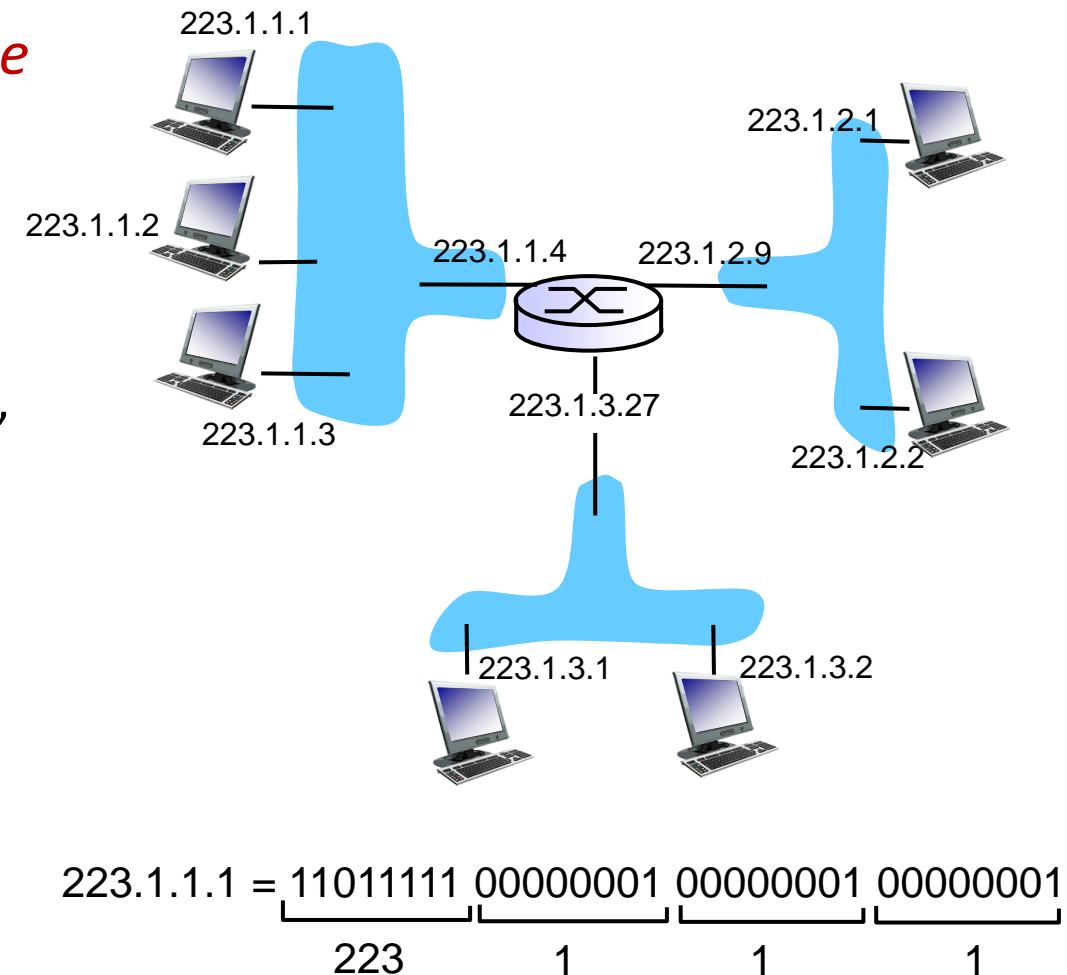
# The Internet network layer

host, router network layer functions:



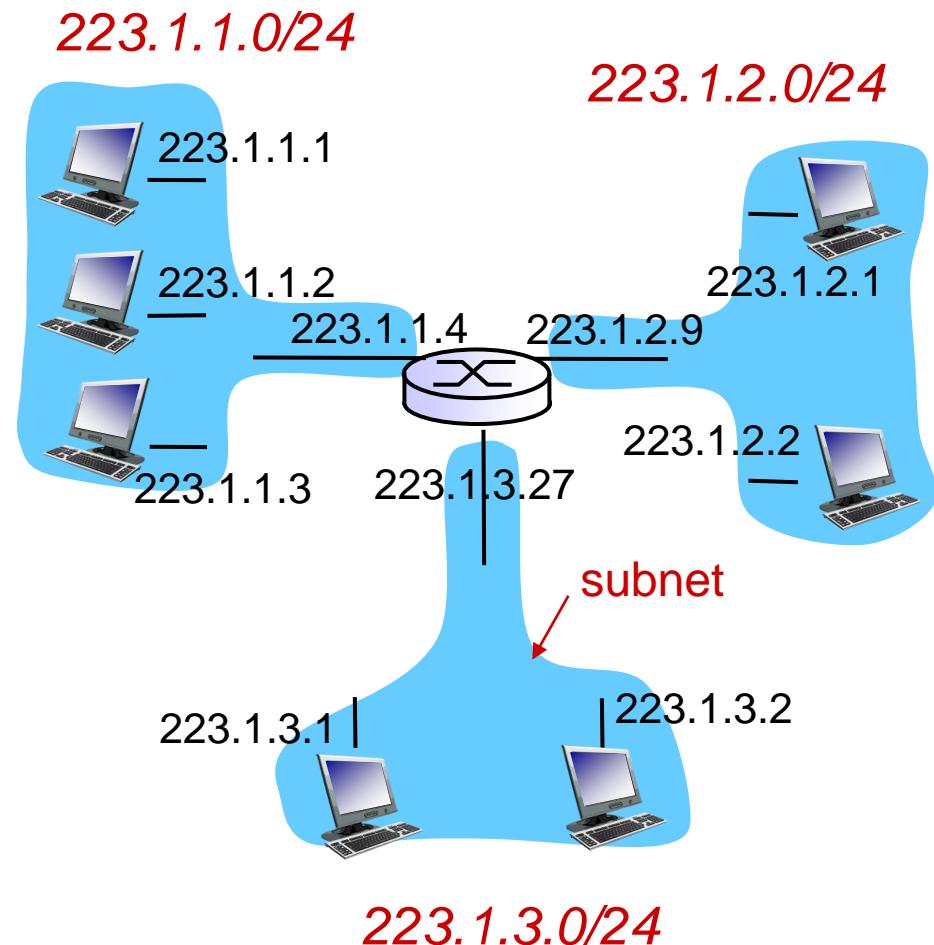
# IP addressing: introduction

- *IP address*: 32-bit id for host/router *interface*
- *interface*: connection between host/router and physical link
  - routers have multiple interfaces
  - common end-host typically has 1-2 interfaces (e.g., wired Ethernet and wireless 802.11); servers can have more



# Subnets

- IP address:
  - subnet part: high order bits (variable number)
  - host part: low order bits
- *what's a subnet ?*
  - Devices that can physically reach each other *without intervening router*
  - device interfaces have same subnet-part (prefix) of IP address



**subnet mask:** eg /24  
defines how to find the subnet part of the address ...

# IP addressing: CIDR

## CIDR: Classless InterDomain Routing

- subnet portion of address of arbitrary length
- address format: **a.b.c.d/x**, where x is # bits in subnet portion of address



# Subnets, masks, calculations

Example subnet: 192.168.5.0/24

|                                                                                                   | Binary form                                                             | Dot-decimal notation |
|---------------------------------------------------------------------------------------------------|-------------------------------------------------------------------------|----------------------|
| IP address                                                                                        | 11000000.10101000.00000101.10000010                                     | 192.168.5.130        |
| Subnet mask                                                                                       | 11111111.11111111.11111111.00000000<br>-----24 first bits set to 1----- | 255.255.255.0        |
| Network prefix:<br><i>bitwise AND of<br/>(address, mask)</i>                                      | 11000000.10101000.00000101.00000000                                     | 192.168.5.0          |
| Host part<br>(obtained with<br>similar calculation,<br>with a mask having<br>the 8 last bits = 1) | 00000000.00000000.00000000.10000010                                     | 0.0.0.130            |

# Longest prefix matching

## longest prefix match

when looking for forwarding table entry for given destination address, use *longest* address prefix that matches destination address.

| Destination Address Range        | Link interface |
|----------------------------------|----------------|
| 11001000 00010111 00010*** ***** | 0              |
| 11001000 00010111 00011000 ***** | 1              |
| 11001000 00010111 00011*** ***** | 2              |
| otherwise                        | 3              |

examples:

11001000 00010111 00010110 10100001 which interface?

11001000 00010111 00011000 10101010 which interface?

# Longest prefix matching

## longest prefix match

when looking for forwarding table entry for given destination address, use *longest* address prefix that matches destination address.

| Destination Address Range        | Link interface |
|----------------------------------|----------------|
| 11001000 00010111 00010*****     | 0              |
| 11001000 00010111 00011000 ***** | 1              |
| 11001000 1 00011*** *****        | 2              |
| otherwise                        | 3              |

examples:

11001000 00010111 00010110 10100001 which interface?

11001000 00010111 00011000 10101010 which interface?

# Longest prefix matching

## longest prefix match

when looking for forwarding table entry for given destination address, use *longest* address prefix that matches destination address.

| Destination Address Range |          |          |       |   | Link interface |
|---------------------------|----------|----------|-------|---|----------------|
| 11001000                  | 00010111 | 00010*** | ***** | * | 0              |
| 11001000                  | 00010111 | 00011000 | ***** | * | 1              |
| 11001000                  | 00010111 | 00011*** | ***** | * | 2              |
| otherwise                 |          |          |       |   | 3              |

match!

examples:

|          |          |          |          |                  |
|----------|----------|----------|----------|------------------|
| 11001000 | 00010111 | 00010110 | 10100001 | which interface? |
| 11001000 | 00010111 | 00011000 | 10101010 | which interface? |

# Longest prefix matching

## longest prefix match

when looking for forwarding table entry for given destination address, use *longest* address prefix that matches destination address.

| Destination Address Range |          |          |       | Link interface |
|---------------------------|----------|----------|-------|----------------|
| 11001000                  | 00010111 | 00010*** | ***** | 0              |
| 11001000                  | 00010111 | 00011000 | ***** | 1              |
| 11001000                  | 00010111 | 00011*** | ***** | 2              |
| otherwise                 |          |          |       | 3              |

match!

examples:

|          |          |          |          |                  |
|----------|----------|----------|----------|------------------|
| 11001000 | 00010111 | 00010110 | 10100001 | which interface? |
| 11001000 | 00010111 | 00011000 | 10101010 | which interface? |

# Roadmap



- Recalling from introduction
  - A day in the life of a web request
- Some random repetition
  - UDP Checksum, Error Detection
  - Classless Inter-Domain Routing
  - Dijkstra's algorithm
  - CSMA
  - CDMA
- Reflections, Perspectives

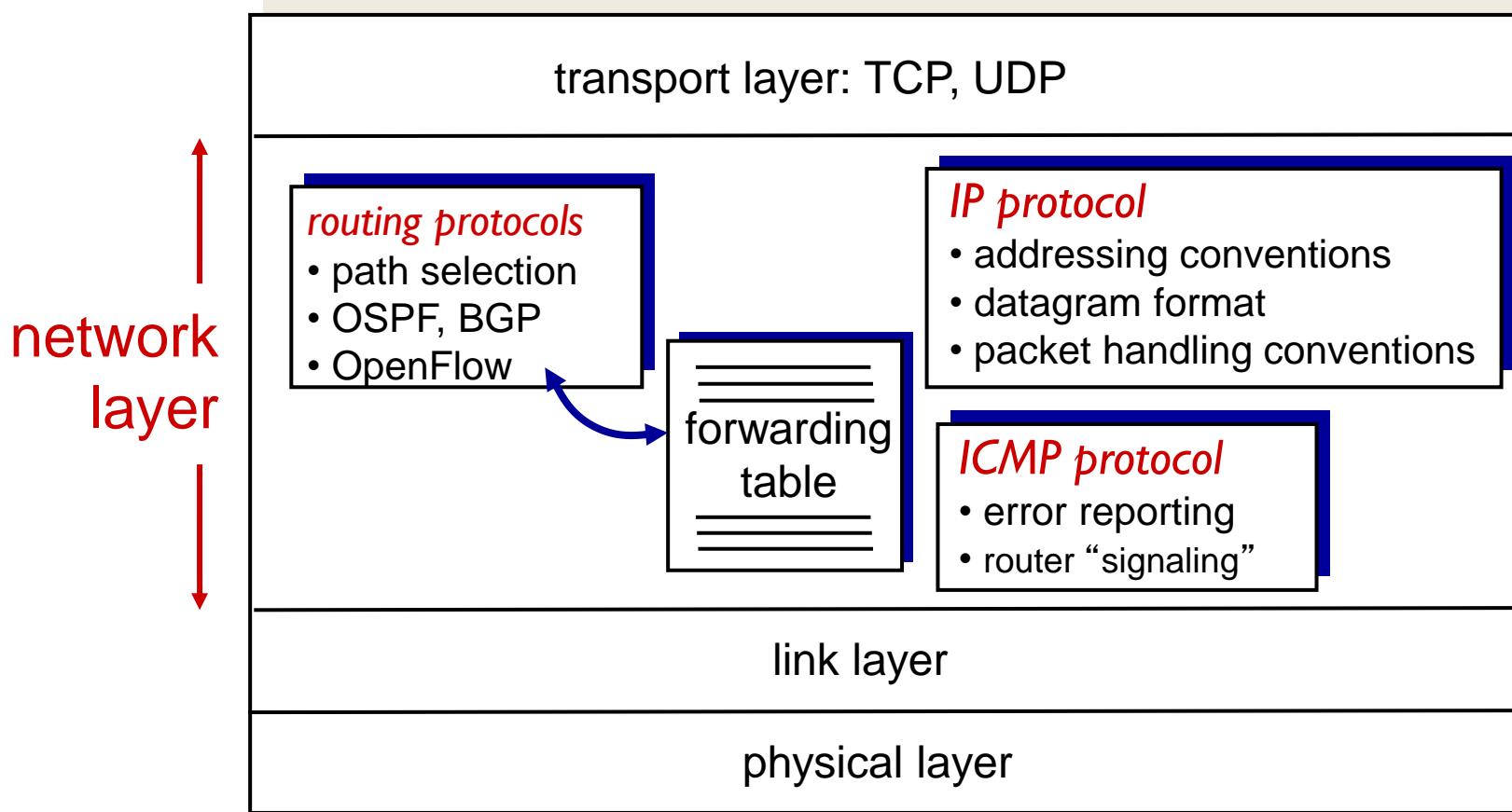
## Request 3: Dijkstra's algorithm

---

I would like you to go through subjects of  
Dijkstra's algorithm to find the shortest path, additionally its tables when it goes through  
edges and nodes.

# The Internet network layer

host, router network layer functions:



# Routing algorithm classification

*How fast  
do routes  
change?*

**static:** routes change  
slowly over time

**global:** all routers have *complete*  
topology, link cost info  
• “link state” algorithms

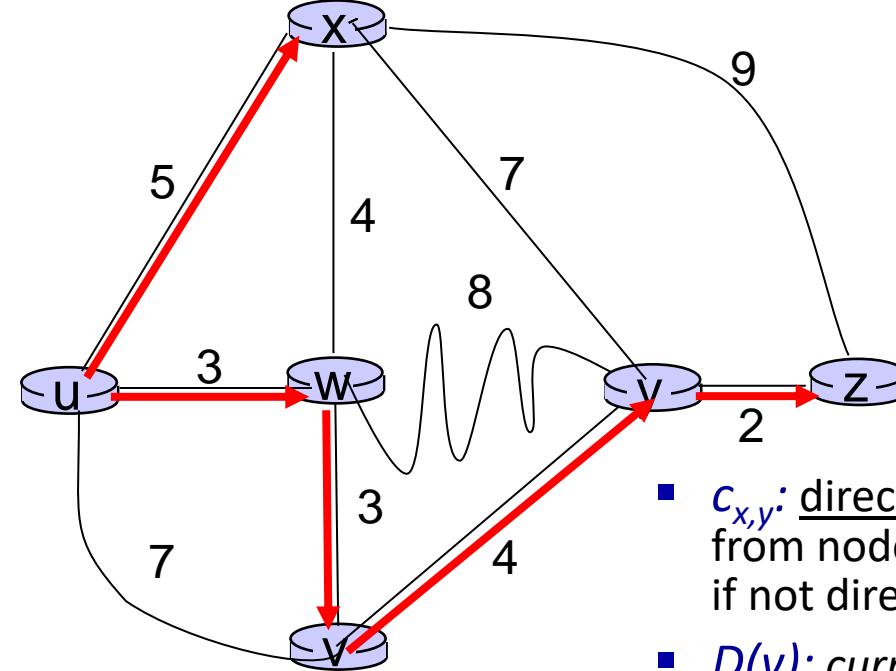
**dynamic:** routes change  
more quickly  
• periodic updates or in  
response to link cost  
changes

**decentralized:** iterative process of  
computation, exchange of info with neighbours  
• routers initially only know link costs to  
attached neighbors  
• “distance vector” algorithms

*global or decentralized information?*

# Dijkstra's algorithm: another example

| Step | $N'$   | $D(v)$ ,<br>$p(v)$ | $D(w)$ ,<br>$p(w)$ | $D(x)$ ,<br>$p(x)$ | $D(y)$ ,<br>$p(y)$ | $D(z)$ ,<br>$p(z)$ |
|------|--------|--------------------|--------------------|--------------------|--------------------|--------------------|
| 0    | u      | 7, u               | 3, u               | 5, u               | $\infty$           | $\infty$           |
| 1    | uw     | 6, w               | 5, u               | 11, w              | $\infty$           |                    |
| 2    | uwx    | 6, w               |                    | 11, w              | 14, x              |                    |
| 3    | uwxv   |                    | 10, v              | 14, x              |                    |                    |
| 4    | uwxvy  |                    |                    | 12, y              |                    |                    |
| 5    | uwxvzy |                    |                    |                    |                    |                    |



Initialization (step 0): For all  $a$ : if  $a$  adjacent to then  $D(a) = c_{u,a}$

find  $a$  not in  $N'$  such that  $D(a)$  is a minimum  
add  $a$  to  $N'$   
update  $D(b)$  for all  $b$  adjacent to  $a$  and not in  $N'$ :  

$$D(b) = \min(D(b), D(a) + c_{a,b})$$

- $c_{x,y}$ : direct link cost from node  $x$  to  $y$ ;  $= \infty$  if not direct neighbors
- $D(v)$ : current estimate of cost of least-cost-path from source to destination  $v$
- $p(v)$ : predecessor node along path from source to  $v$
- $N'$ : set of nodes whose least-cost-path definitively known

# Roadmap



- Recalling from introduction
  - A day in the life of a web request
- Some random repetition
  - UDP Checksum, Error Detection
  - Classless Inter-Domain Routing
  - Dijkstra's algorithm
  - CSMA
  - CDMA
- Reflections, Perspectives

# Request 3: Dijkstra's algorithm

---

I would like you to go through subjects of

The word "CSMA" is rendered in a pixelated, blocky font, appearing to float in the air. The letters are composed of small colored squares, giving them a digital or retro video game aesthetic.

# Layering: The OSI Reference Model

ISO (International Standards Organization) defined the OSI (Open Systems Interconnect) model to help vendors create interoperable network implementation

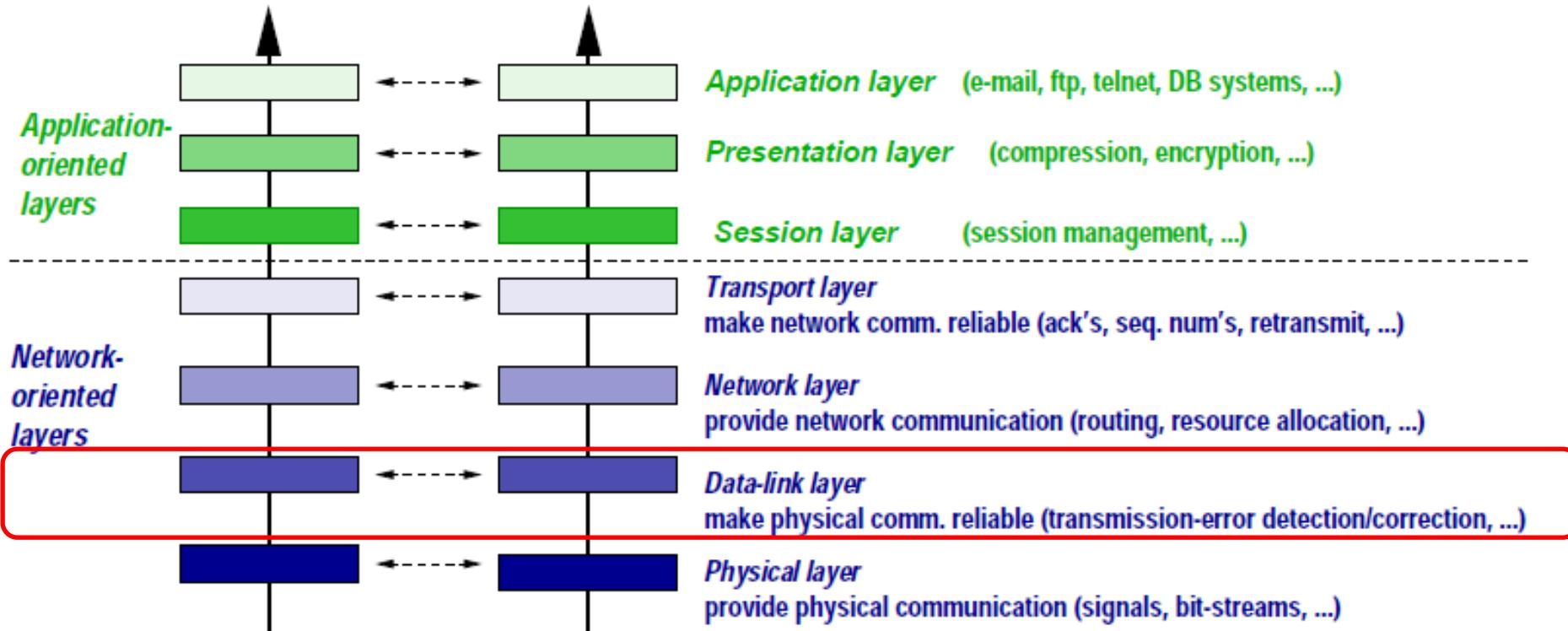


Fig. Steen, Sips : Computer and Network organization

``X dot'' series (X.25, X. 400, X.500) OSI model implementation (protocol stack)

# CSMA (carrier sense multiple access)

---

Simple **CSMA**: listen before transmit:

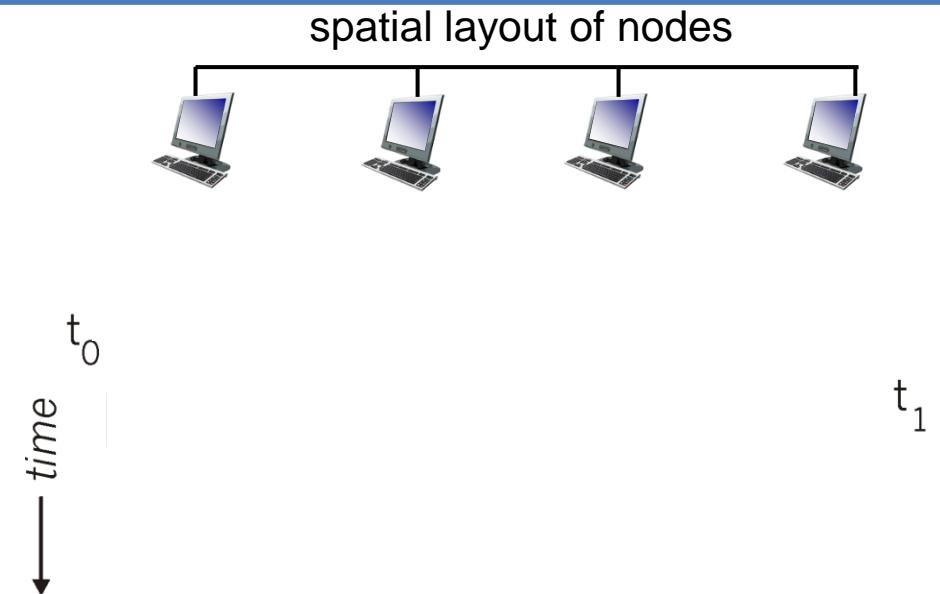
- if channel sensed idle: transmit entire frame
- if channel sensed busy: defer transmission
- Human analogy: **don't interrupt others!**

**CSMA/CD**: CSMA with *collision detection*

- Collisions *detected* within short time
- Colliding transmissions aborted, reducing channel wastage
- Collision detection easy in wired, difficult with wireless
- Human analogy: the polite conversationalist

# CSMA: collisions

- Collisions *can* still occur with carrier sensing:
  - Propagation delay means two nodes may not hear each other's just-started transmission
- Collision: entire packet transmission time wasted
  - Distance & propagation delay play role in determining collision probability
- CSMA/CD reduces the amount of time wasted in collisions
  - transmission aborted on collision detection



# Ethernet CSMA/CD algorithm

---

1. NIC receives datagram from network layer, creates frame
2. If NIC senses channel:
  - if **idle**: start frame transmission.
  - if **busy**: wait until channel idle, then transmit
3. If NIC transmits entire frame without collision, NIC is done with frame !
4. If NIC detects another transmission while sending: abort, send jam signal
5. After aborting, NIC enters ***binary (exponential) backoff***:
  - after  $m$ th collision, NIC chooses  $K$  at random from  $\{0,1,2, \dots, 2^m-1\}$ . NIC waits  $K \cdot 512$  bit times, returns to Step 2
  - more collisions: longer backoff interval

# Roadmap



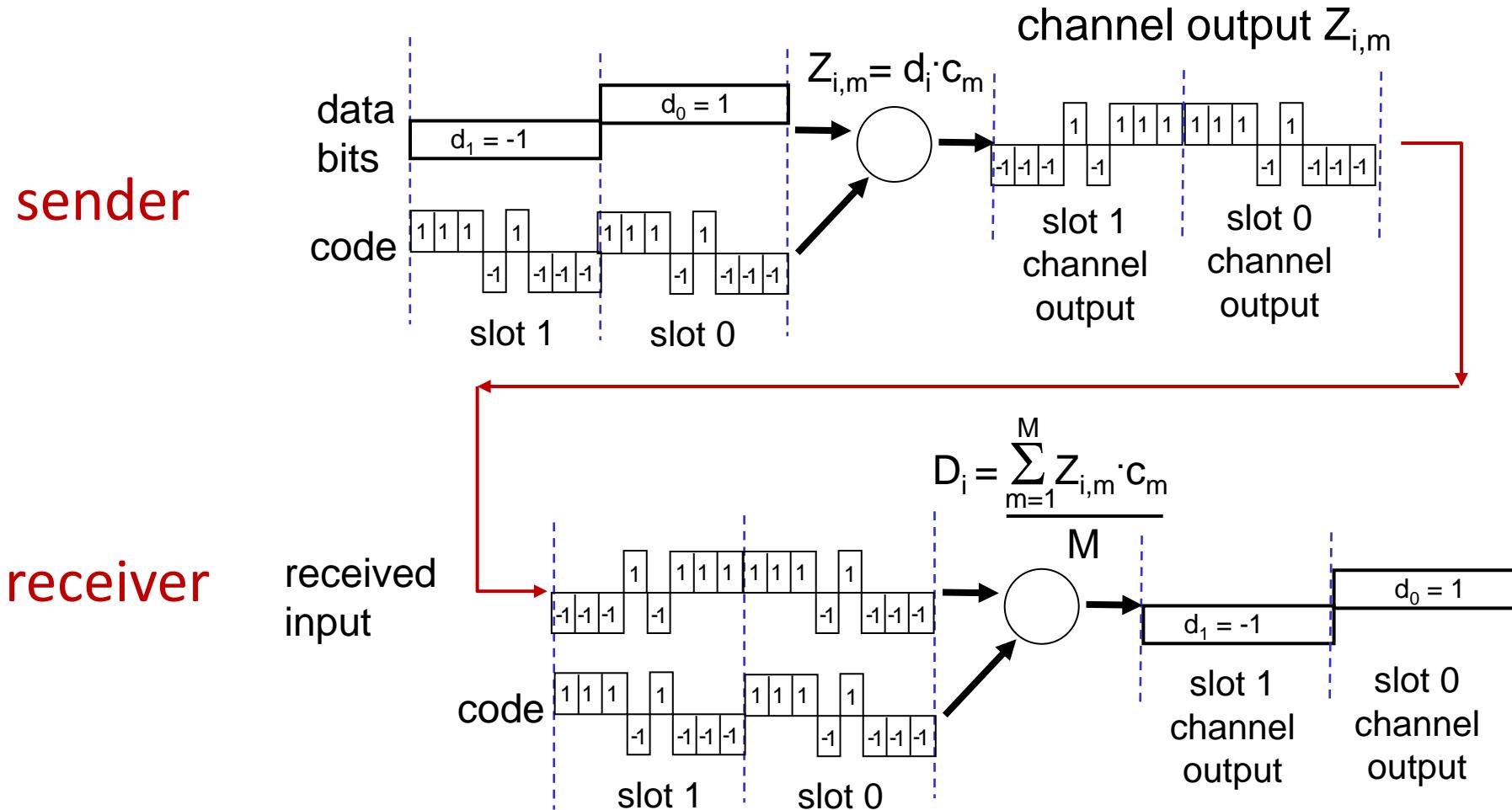
- Recalling from introduction
  - A day in the life of a web request
- Some random repetition
  - UDP Checksum, Error Detection
  - Classless Inter-Domain Routing
  - Dijkstra's algorithm
  - CSMA
  - CDMA
- Reflections, Perspectives

# Code Division Multiple Access (CDMA)

---

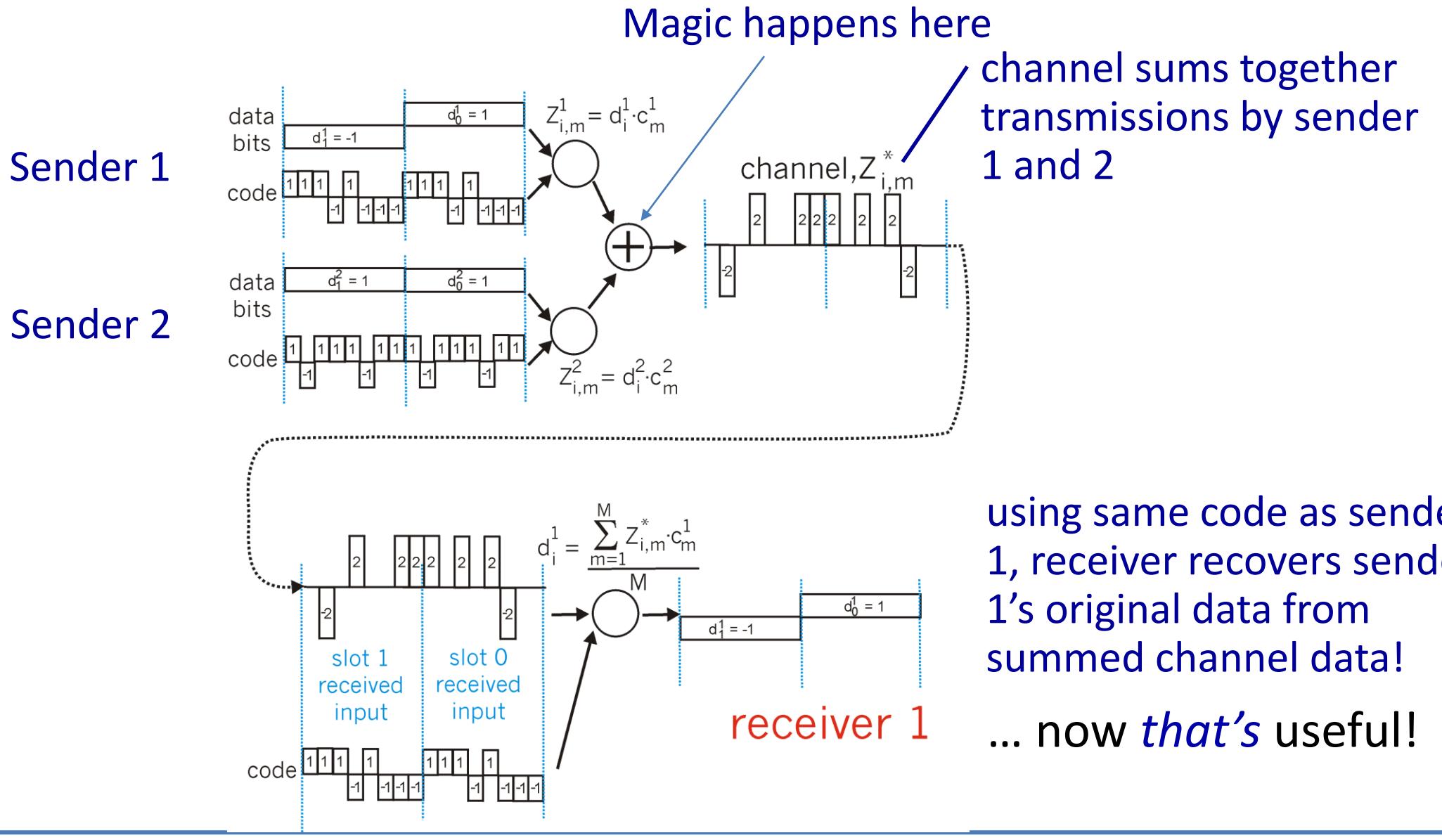
- CDMA is a channel access method such that **several transmitters can send information at the same time over a single communication channel.**
- Unique “code” assigned to each user; i.e., code set partitioning
  - All users share same frequency, but each user has own “chipping” sequence (i.e., code) to encode data.
  - Allows multiple users to “coexist” and transmit simultaneously with minimal interference (if codes are “orthogonal”).
- **Encoding:** inner product: (original data)  $\times$  (chipping sequence)
- **Decoding:** summed inner-product: (encoded data)  $\times$  (chipping sequence)

# CDMA encode/decode



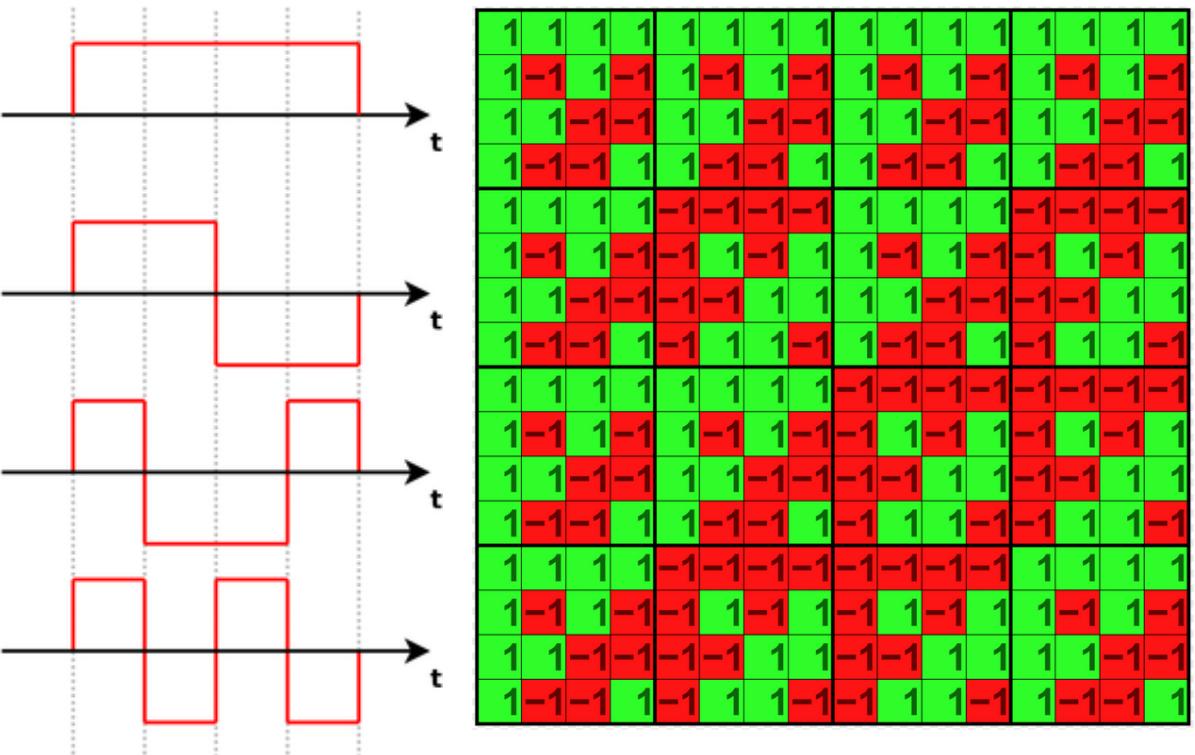
... but this is not really useful yet, is it?

# CDMA encode/decode magic



# Orthogonal codes

- **Encoding:** inner product: (original data)  $X$  (chipping sequence)
- **Decoding:** summed inner-product: (encoded data)  $X$  (chipping sequence)
- The chipping sequences must be orthogonal so that decoding can be successful for several clients.
  - Two elements  $u, v$  (of a vector space) with a bilinear form  $B$  are orthogonal when  $B(u, v) = 0$ .
    - In English: There is some function that linearly and for each argument separately “makes something zero”.
    - The dot product on  $\mathbb{R}^n$  is (conveniently) an example of a bilinear form.



- The rows and columns of the **Welsh matrix** are orthogonal.
  - Their dot product is always zero.
- The (rows of the) Welsh matrix provide us with chipping sequences.

# Roadmap



- Recalling from introduction
  - A day in the life of a web request
- Some random repetition
  - UDP Checksum, Error Detection
  - Classless Inter-Domain Routing
  - Dijkstra's algorithm
  - CSMA
  - CDMA
- Reflections, Perspectives



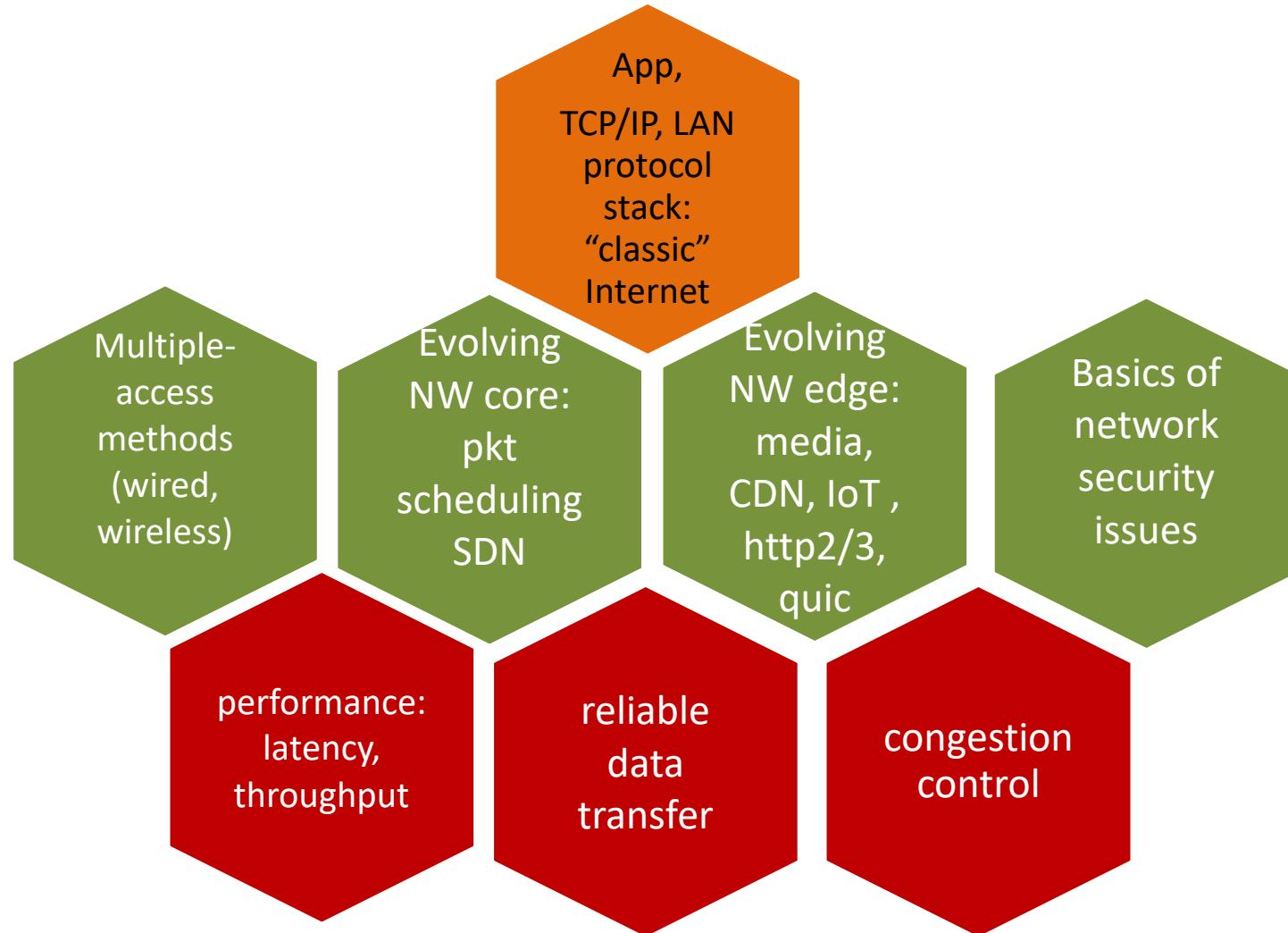
# Roadmap



- Recalling from introduction
  - A day in the life of a web request
- Some random repetition
  - UDP Checksum, Error Detection
  - Classless Inter-Domain Routing
  - Dijkstra's algorithm
  - CSMA
  - CDMA
- Reflections, Perspectives



# Cross-Layer view



# Architectural Principles of the Internet

RFC 1958

“Many members of the Internet community would argue that there is no architecture, but only a tradition, which was not written down for the first 25 years (or at least not by the IAB<sup>(\*)</sup>). However, in very general terms, the community believes that

**The goal is connectivity, the tool is the Internet Protocol, and the intelligence is end to end rather than hidden in the network.”**

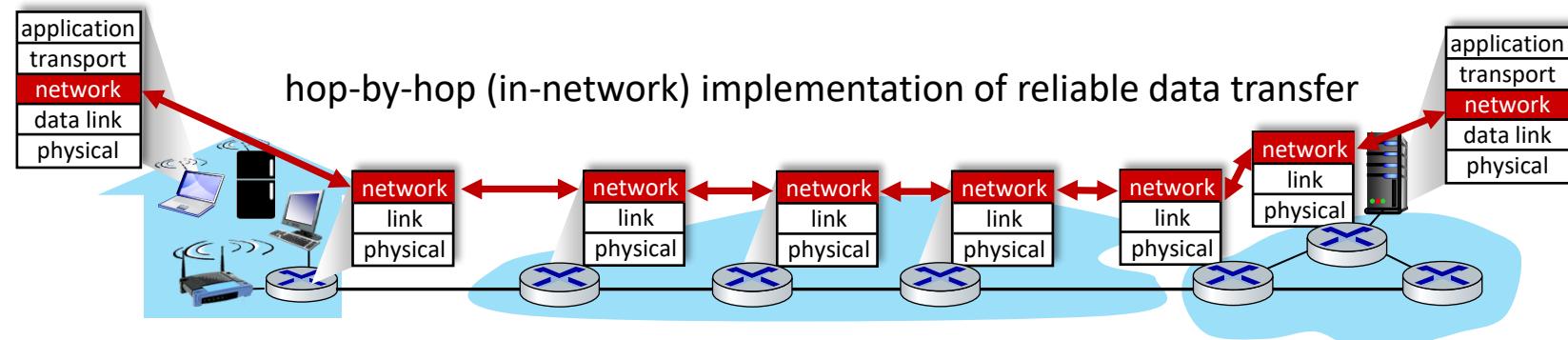
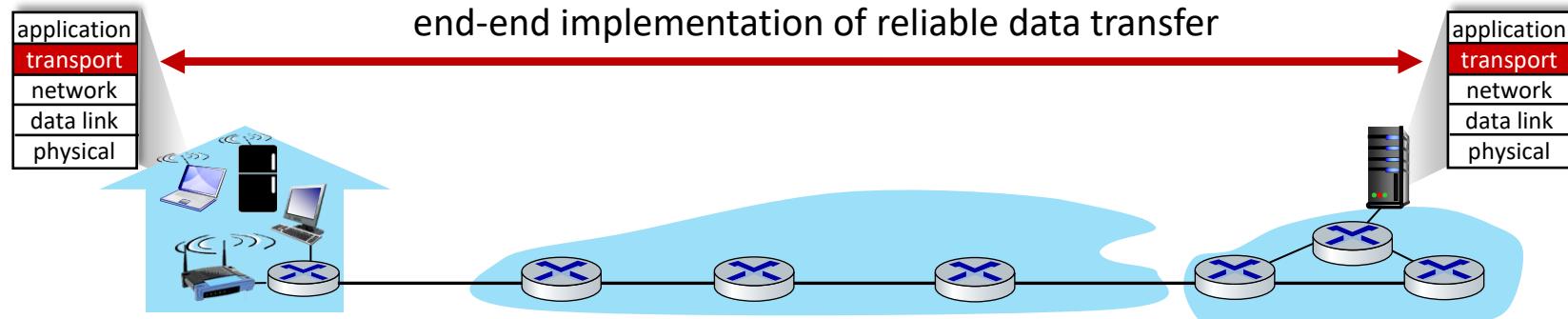
Three cornerstone aspects:

- simple connectivity
- IP protocol: that narrow waist
- intelligence, complexity at network edge

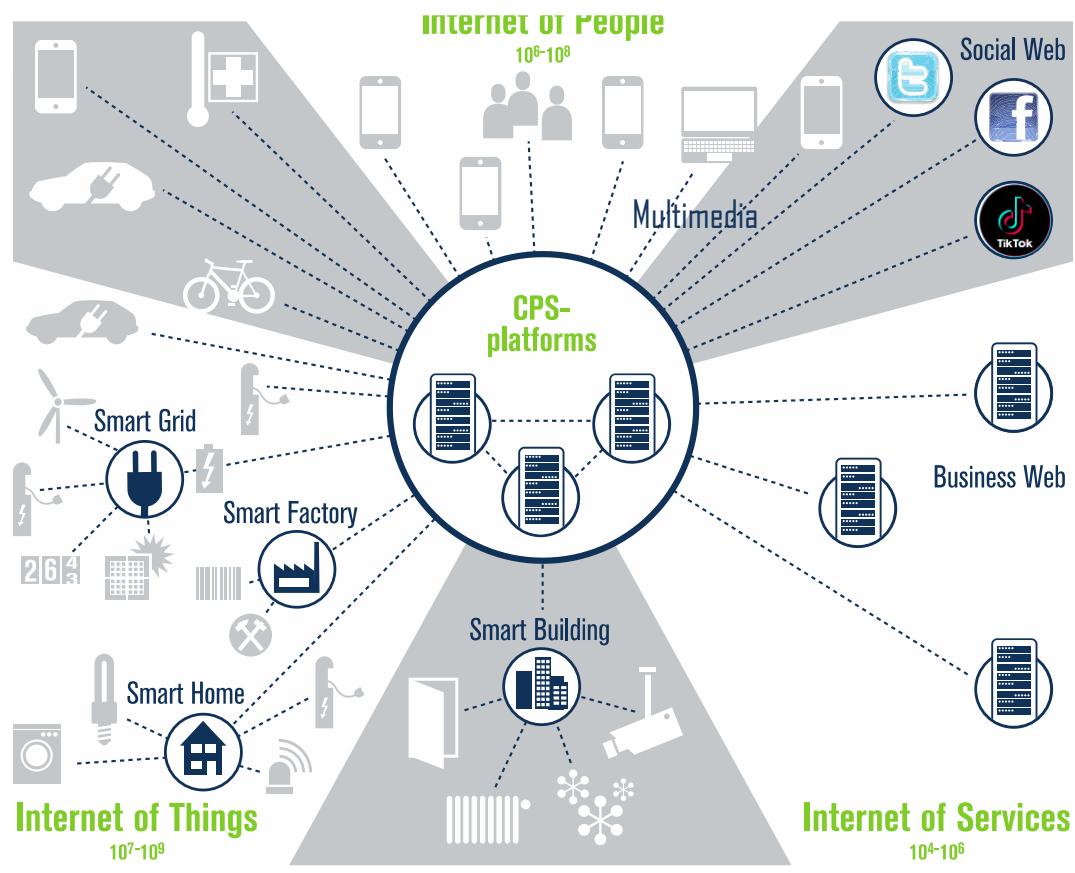
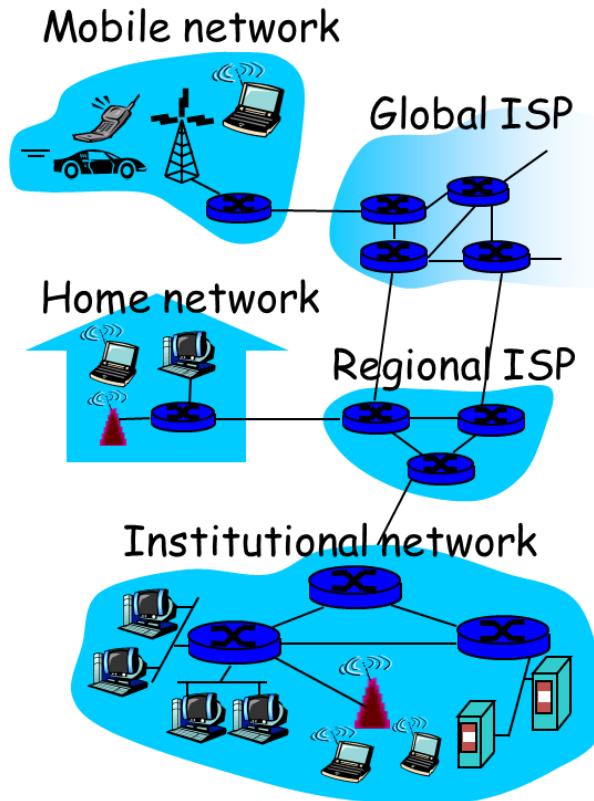
<sup>(\*)</sup> Internet Architecture Board

# The end-end argument

- Some network functionality (e.g., reliable data transfer, congestion) can be implemented in **network**, or at **network edge**



# Internet & its context is changing ....

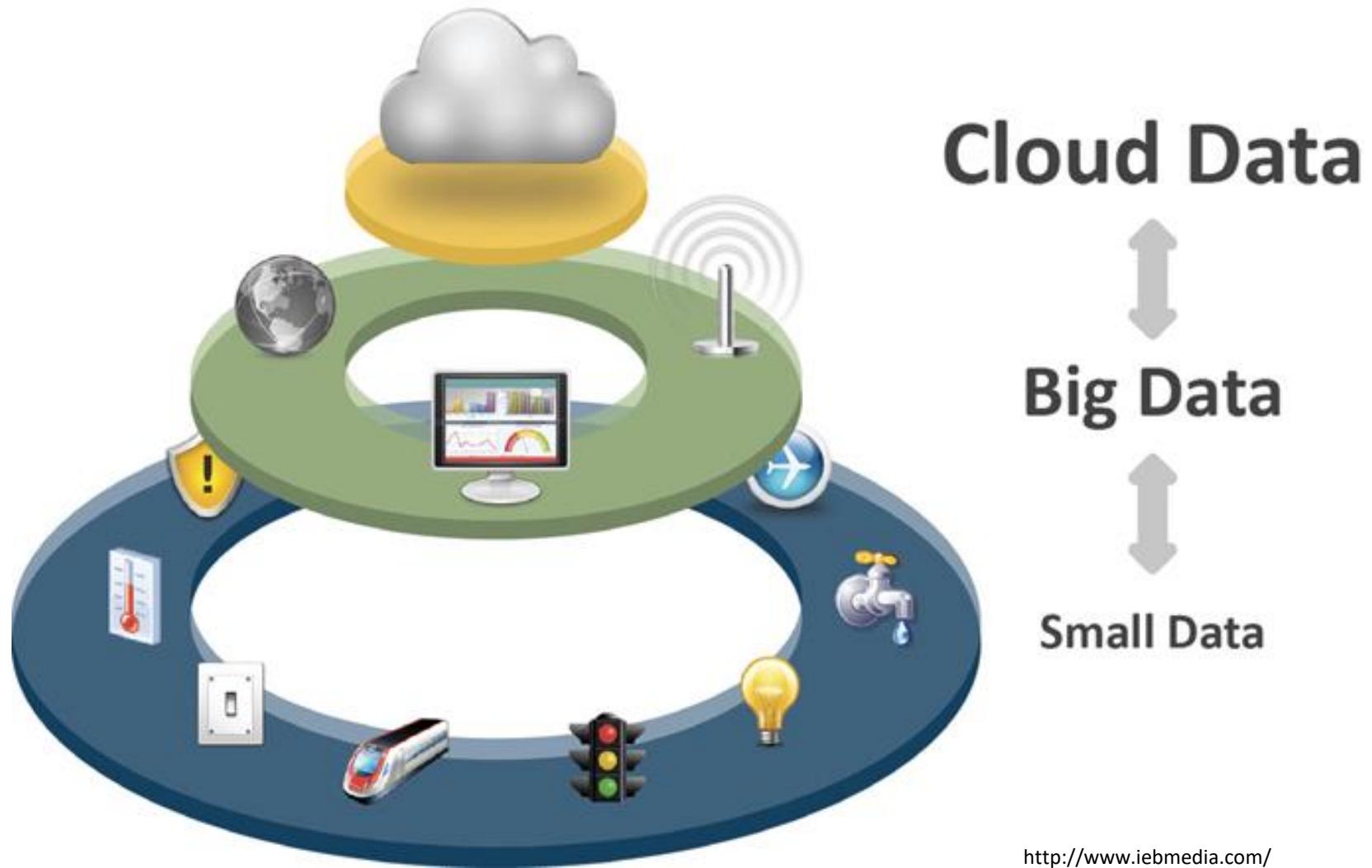


Source: Bosch Software Innovations 2012

continuous evolution ....

# "IoX" - Data processing/ML - Distributed Computing in interplay

A lot of data to be communicated and processed



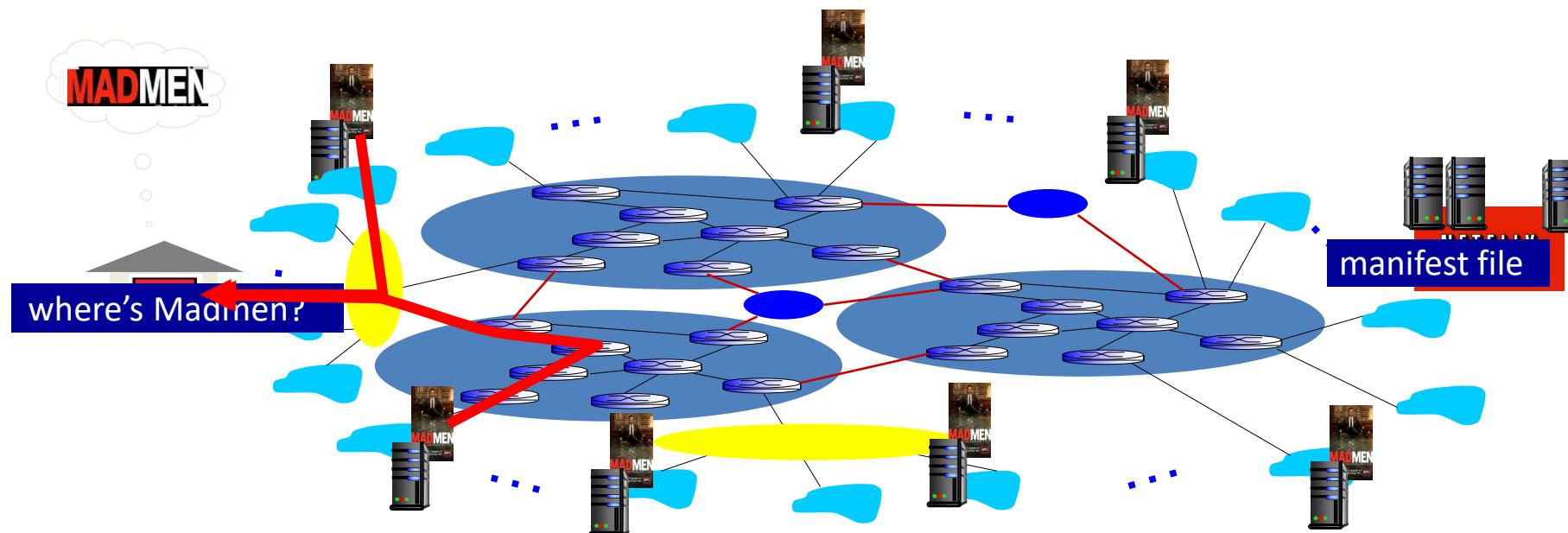
<http://www.iebmedia.com/>

Continuous change @ NW edge:  
p2p, media apps, followed by...

# Content Distribution Networks (CDNs)

- CDN: stores copies of content at CDN nodes
  - e.g. Netflix stores copies of MadMen
- subscriber requests content from CDN
  - directed to nearby copy, retrieves content
  - may choose different copy *if network path congested*

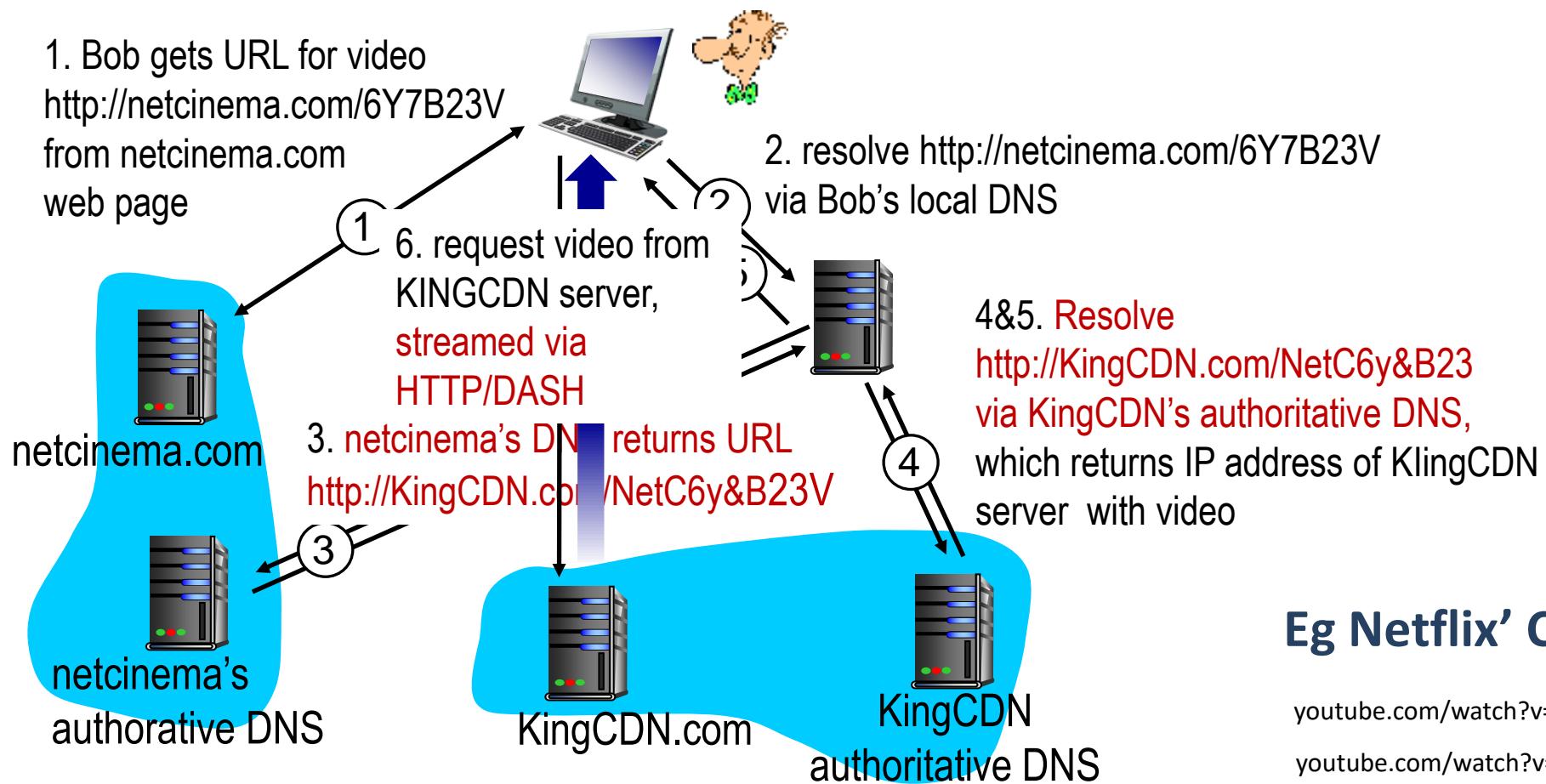
Network-supported  
(DNS++) distributed  
systems/applications



# CDN: “simple” content access scenario

Bob (client) requests video <http://netcinema.com/6Y7B23V>

- video stored in CDN at <http://KingCDN.com/NetC6y&B23V>

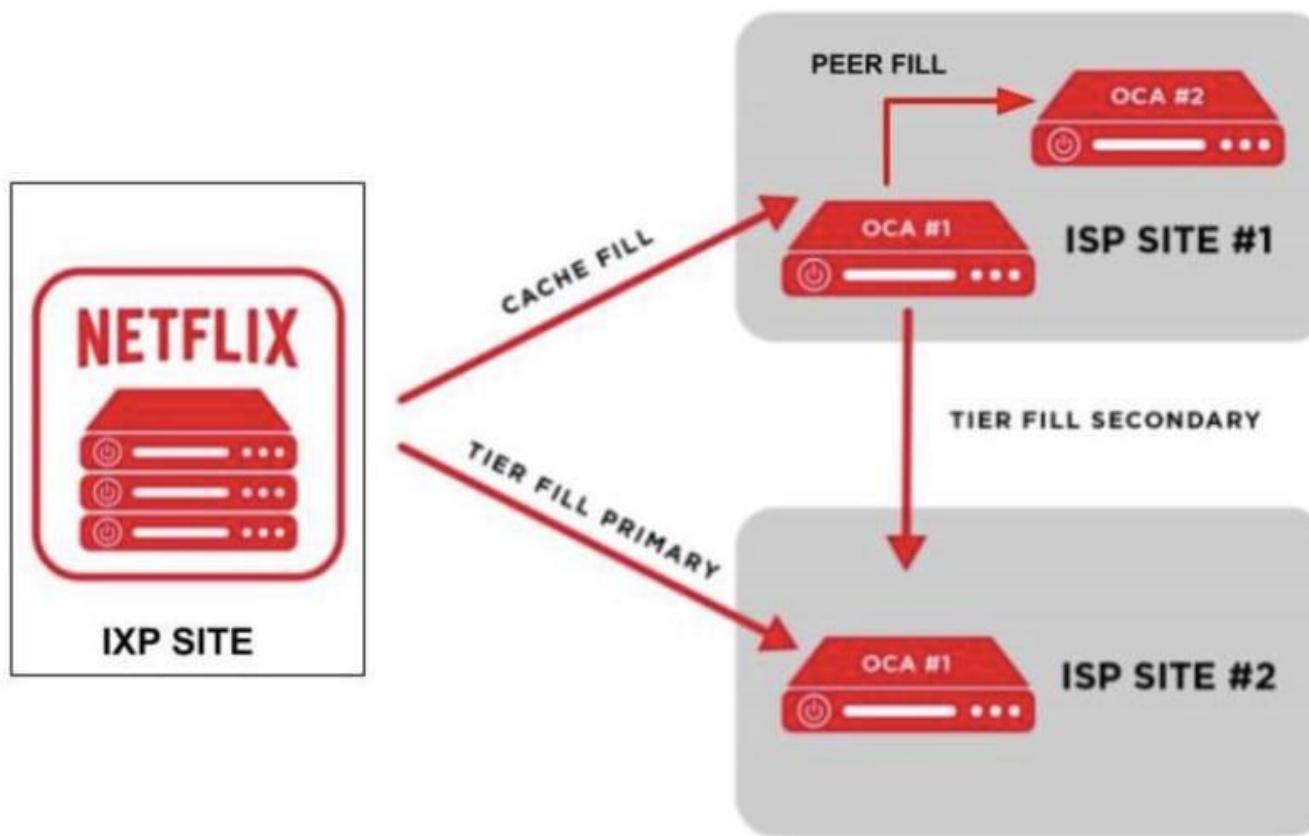


Eg Netflix' CDN approach

[youtube.com/watch?v=LkLLpYdDINA](http://youtube.com/watch?v=LkLLpYdDINA)

[youtube.com/watch?v=tbqcsHg-Q\\_o](http://youtube.com/watch?v=tbqcsHg-Q_o)

# Example of caching: Netflix Open Connect

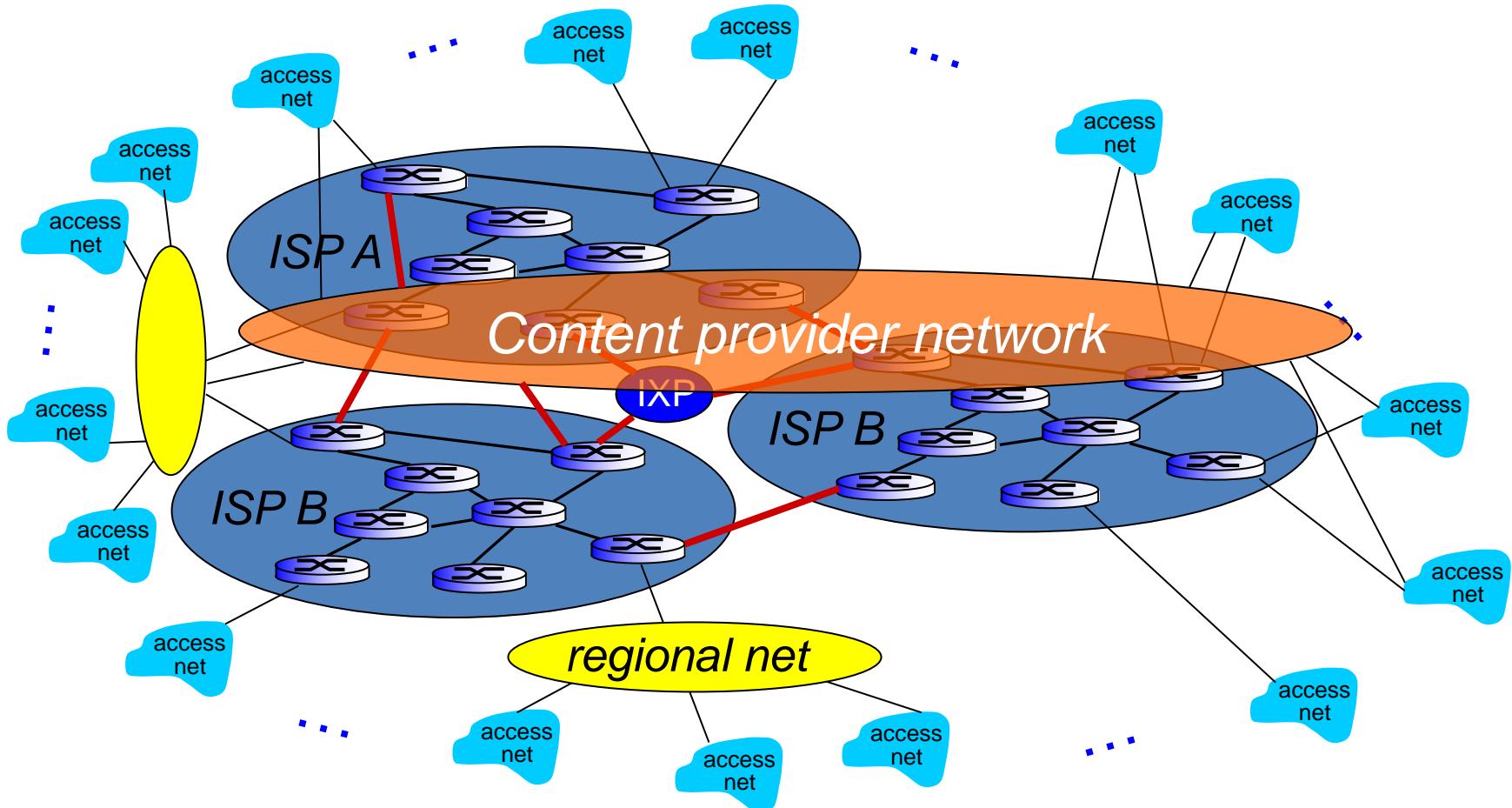


## Example Netflix

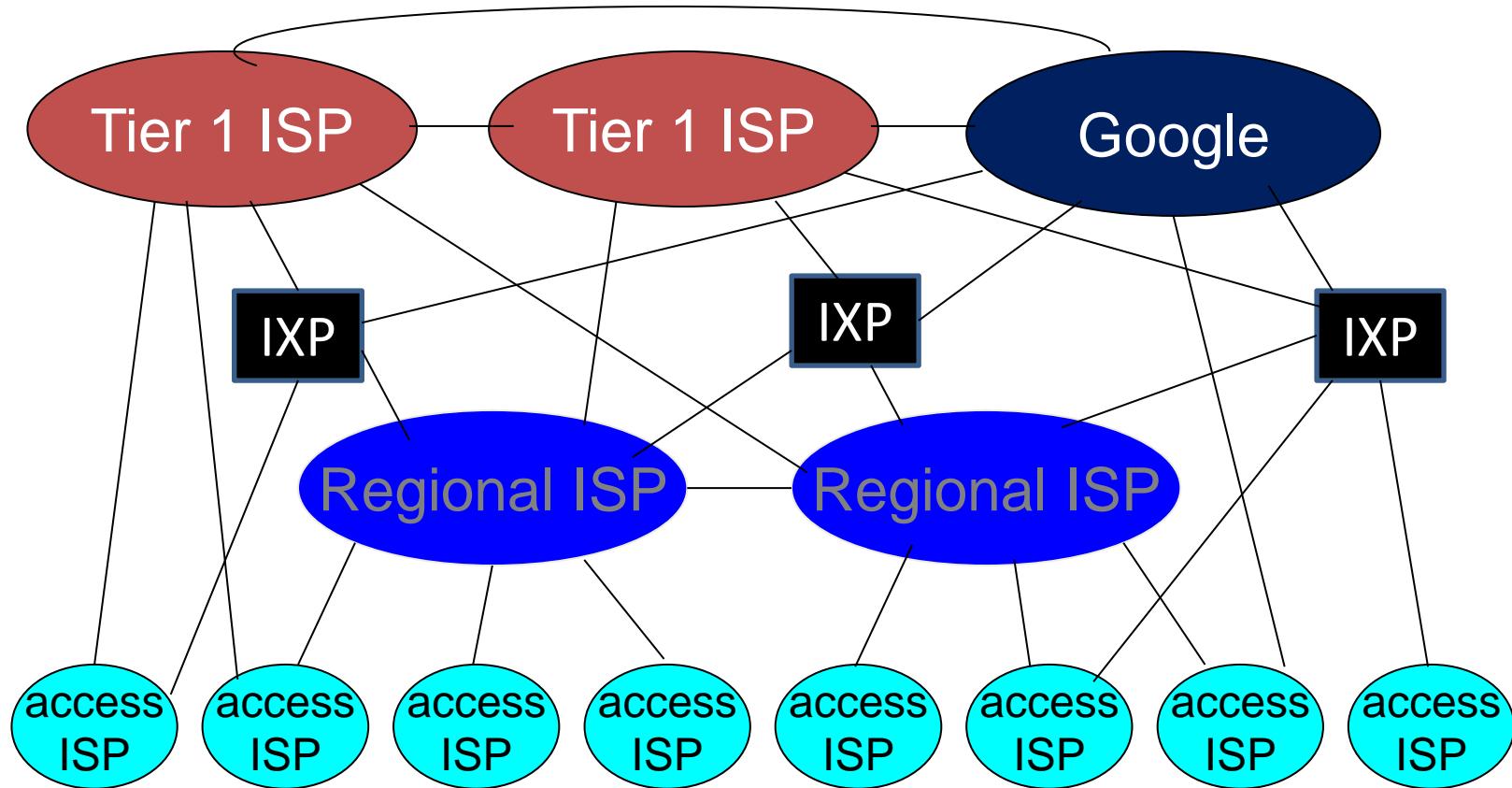
- Netflix splits video files into small chunks
- Popular videos are stored closed to the end-user
- ISP voluntarily install “Netflix open connect” servers to reduce remote access.
- During off-peak periods, content is moved between the open connect servers

# Internet structure: network of networks

... and content provider networks (e.g., Google, Microsoft, Amazon ) run their own network, to bring services, content close to end users



# Internet structure: network of networks



# Data center networks

- 10's to 100's of thousands of hosts, often closely coupled, in close proximity:
  - e-business (e.g. Amazon)
  - content-servers (e.g., YouTube, Akamai, Apple, Microsoft)
  - search engines, data mining (e.g., Google)
  - ...

## ❖ challenges:

- multiple applications, each serving massive numbers of clients
- managing/balancing load, networking, data bottlenecks



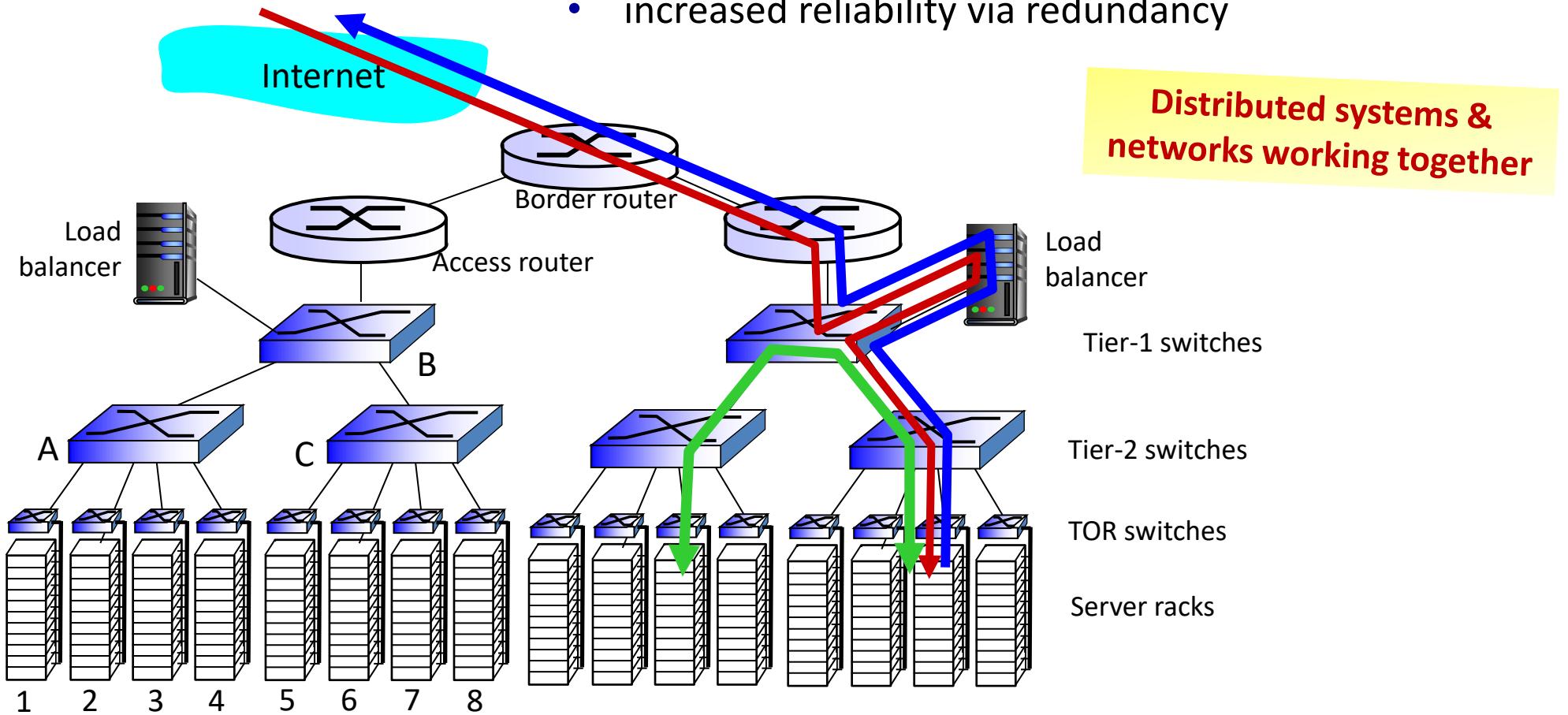
Inside a 40-ft Microsoft container,  
Chicago data center

# Data center networks

load balancer: application-layer routing

- directs client requests (workload) within data center

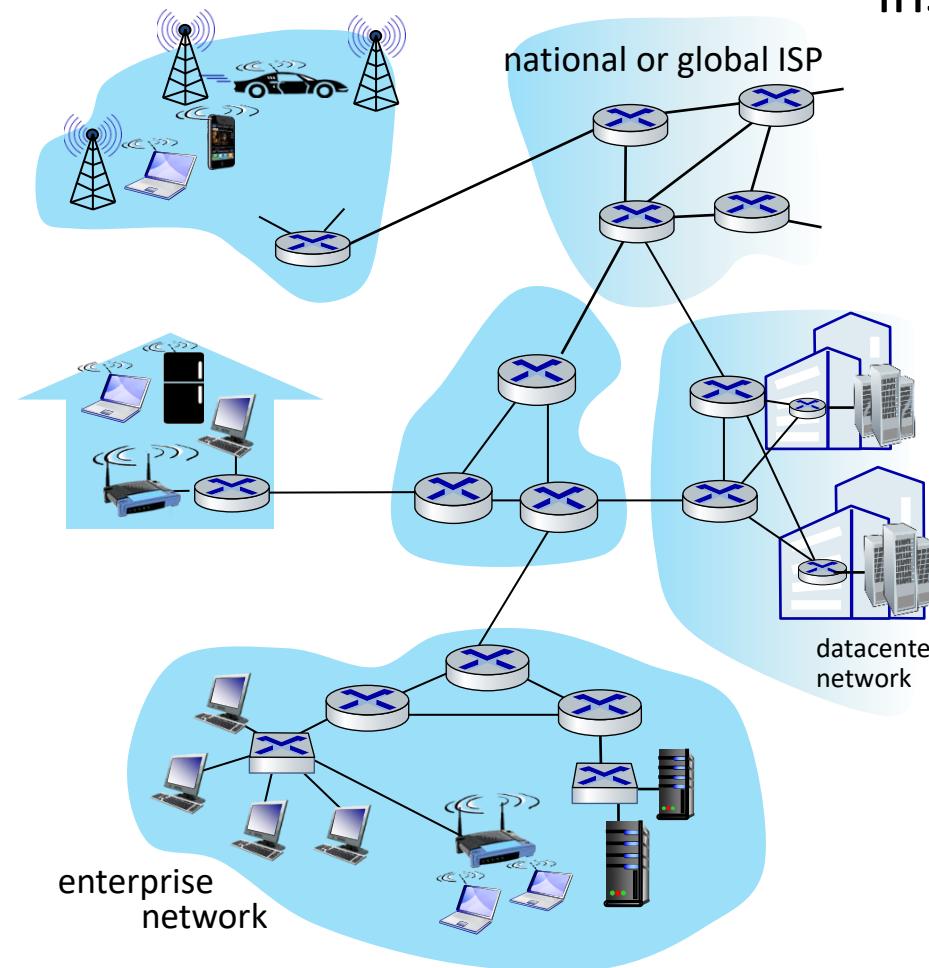
- Rich interconnection among switches, racks:
  - increased throughput between racks (multiple routing paths possible)
  - increased reliability via redundancy



# Middleboxes everywhere!

NAT: home, cellular, institutional

Application-specific: service providers, institutional, CDN

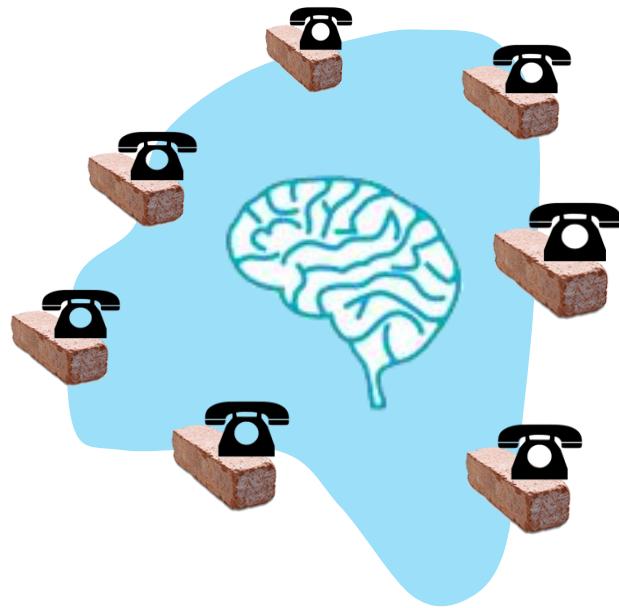


Firewalls, IDS: corporate, institutional, service providers, ISPs

Load balancers: corporate, service provider, data center, mobile nets

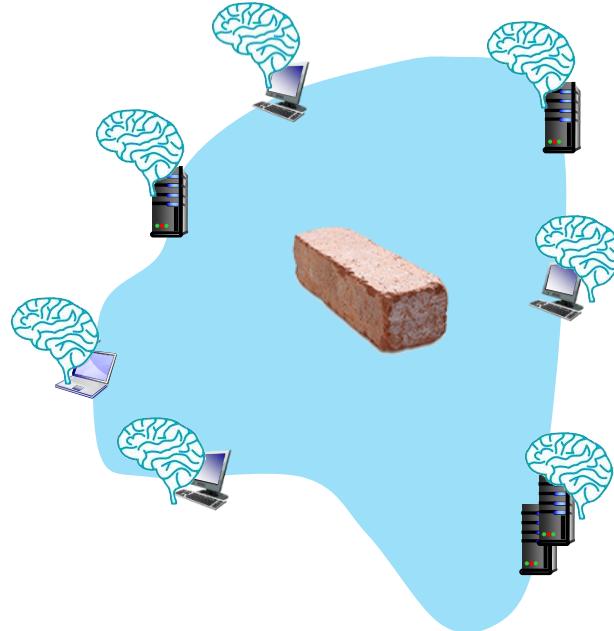
Caches: service provider, mobile, CDNs

# Where's the intelligence?



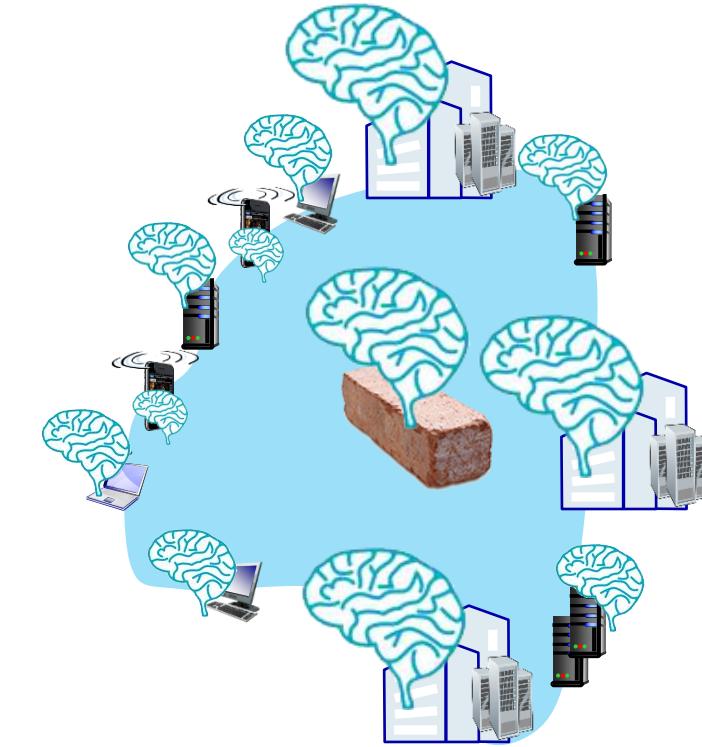
**20<sup>th</sup> century phone net:**

- intelligence/computing at network switches



**Internet (pre-2005)**

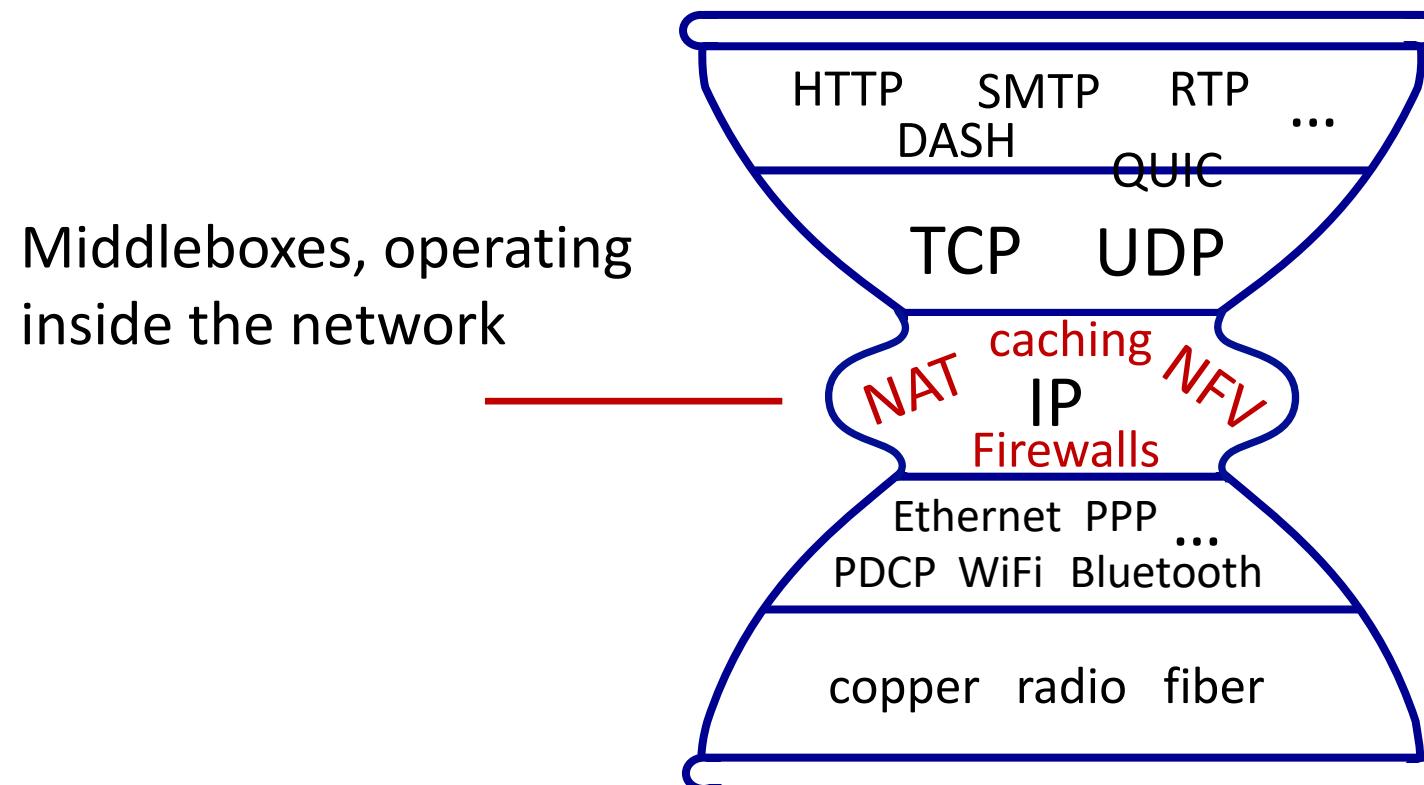
- intelligence, computing at edge



**Internet (post-2005)**

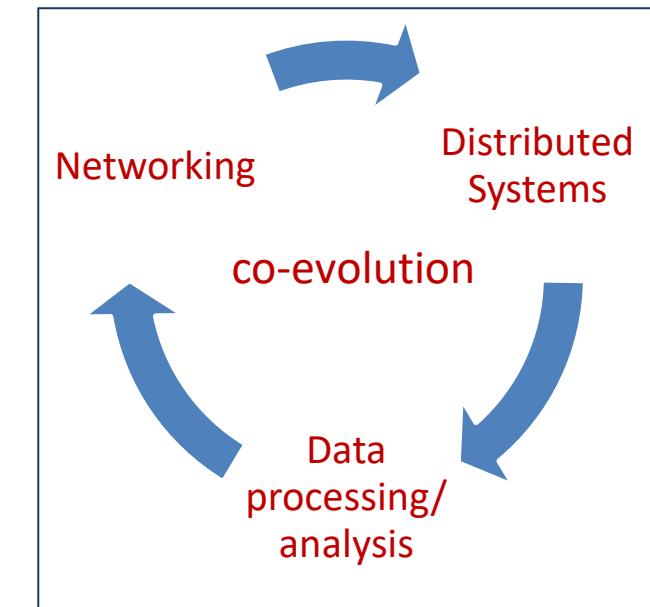
- programmable network devices
- intelligence, computing, massive application-level infrastructure at edge

# The IP hourglass, at middle age 😊



## New possibilities / ongoing work

- 5G for mission-critical communication
- In-NW processing (NW compute fabric)
- Smart NICs as accelerators/enablers
- Data pipelines



# Thank you & best wishes!

**Recall, important for the exam:**

**When/where:** Wednesday March 13, 14.00-18.00, Johanneberg



## To think

In-depth & overview; critical eye; explain; ask yourselves: why is this so? / How does it work? Explain your answers in the exam!

*"If you hear a voice within you say  
'you cannot paint,' then by all means  
paint, and that voice will be  
silenced."* – Vincent Van Gogh

*Good luck with all your efforts!!!*